Cell Assistant for Windows™

Version 1.3

User's Manual

Revised: October 20, 2000

Item #: MANCAUMAV130001
© 1996-2009 Dyne Systems Co., LLC
Portions © 1992-1997 Summit Software Co.

Dyne Systems, Inc.

Contact Information

Address:
W209 N17391 Industrial Drive
Jackson, WI 53037
U.S.A.

Toll-Free Phone:
(800) 657-0726

Fax:
(262) 677-9308

Website:
www.dynesystems.com

Table of Contents

CHAPTER 1 - INTRODUCTION 1-1

Conventions Used in this Manual 1-1
Technical Support 1-1
Feedback 1-1
Overview of Cell Assistant 1-1

CHAPTER 2 - INSTALLATION 2-1

Operating System Requirements 2-1
Hardware Requirements 2-1
Computer Setup 2-1
Installation of Cell Assistant Software 2-1
Installation of Copy Protect Block 2-2

CHAPTER 3 - GETTING STARTED 3-1

TestSystems 3-1
Password Protection 3-2
Main Menu Bar 3-2
Setting System Preferences 3-4
Tool Bar 3-8
Status Bar 3-8

CHAPTER 4 - HARDWARE DEVICES 4-1

Adding a Hardware Device 4-1
Removing a Hardware Device 4-1
Hardware Device Setup 4-1

CHAPTER 5 - CHANNELS 5-1

Channel List 5-1
Display Filter 5-1
Adding Channels 5-1
Removing Channels 5-2
Renaming Channels 5-2
Clipping Channel Names 5-2
Configuring Channels 5-3
Channel Types 5-4

CHAPTER 6 - LIMITS 6-1

Creating a New Limit 6-1
Monitored Channel 6-1
Limit Type 6-1
Trigger Type 6-2
Priority 6-2
Procedure 6-2
Using Limits in TestPlans 6-3
The DSBasic Limit handler 6-4
Limit Sets 6-6

CHAPTER 7 - TESTPLANS 7-1

About DSBasic 7-1
TestPlan Files window 7-1
TestPlan Notes window 7-1
Add button 7-1
Modify button 7-1
Rename button 7-1
Remove button 7-2
Run Button 7-2

Set Default button 7-2
Edit Global button 7-2
Properties button 7-3
TestPlan Programming Topics 7-4

CHAPTER 8 - DATA FILES 8-1

Adding Data File Formats 8-1
Renaming Data File Formats 8-1
Removing Data File Formats 8-1
Configuring Data File Formats 8-1
Using Data Files in TestPlans 8-2
Sharing Data Files with other applications 8-7

CHAPTER 9 - CALIBRATION 9-1

1 Point Calibration 9-1
2 Point Calibration 9-2
Manual Calibration 9-2
Saving Calibrations 9-2
Restoring Calibrations 9-3

CHAPTER 10 - VIRTUAL INSTRUMENT PANELS 10-1

Using VIP's in Cell Assistant 10-1
Creating VIP's 10-2
VIPCONTROLS ActiveX Control 10-3

CHAPTER 11 - UTILITIES 11-1

Channel Viewer Utility 11-1
CSV File Viewer Utility 11-1
Limit Set Viewer Utility 11-2

CHAPTER 12 - TROUBLESHOOTING 12-1

Device Communications Error Dialog Boxes 12-1
Failures When Loading a TestPlan 12-1
Watchdog Trips Occur 12-1
CSV File Viewer Cannot Load Entire Data File 12-2
Getting VIP Errors From the CADAQ Control 12-2
Problems Associated with NT User Administration 12-3
New Device is Detected Whenever Computer is Restarted 12-4
Communcation Errors from the Dyn-Loc Diagnostic Page 12-4
Sharing Hardware Devices with Other Computers 12-4
Dr. Watson 12-5

APPENDIX A - DEVICE CHANNELS A-1

APPENDIX B - DSBASIC LANGUAGE EXTENSIONS B-1

APPENDIX C - COMPUTER SETUP INSTRUCTIONS C-1

APPENDIX D - CADAQ OLE CONTROL D-1

INDEX

Chapter 1 - Introduction

Conventions Used in this Manual

Bold Italic type will be used to refer to text that will actually be seen on the monitor. Examples include ***OK*** button and ***Channels*** dialog box.

Technical Support

Please feel free to call DyneSystemsCo. at 1-800-657-0726 with any questions that you may have. You have made a substantial investment and we are anxious to help you get the maximum value from your Cell Assistant System.

In addition, we want the manual to be as helpful as possible. Please feel free to call with suggestions, or simply copy the manual page, mark your suggestions (or, in very rare cases, your corrections), and FAX them to Jeff Bert at 1-414-250-2710.

Feedback

Please feel free to contact us with any comments you have on this manual. We want it to be as helpful as possible.

Overview of Cell Assistant

Cell Assistant is a very powerful data acquisition and control system. Data is continuously acquired (i.e. sampled) from all configured devices. Cell Assistant provides drivers for Dynamometer controllers, Programmable Logic Controllers (PLC's), gas analyzers, fuel systems, etc … Once a driver is installed and configured, each presents a unique set of data channels that can be read by the Cell Assistant data acquisition subsystem. Each driver also presents data channels that can be written to by the Cell Assistant control subsystem.

The control subsystem consists of a running TestPlan that is written in the DSBasic programming language. DSBasic is very similar to the Microsoft Visual Basic programming language except that it incorporates many language extensions that are useful for data acquisition and control. The running program has access to all data that is being sampled by the data acquisition subsystem. Also the running program can easily send commands to any device using the DSBasic language extensions mentioned above. DSBasic is an excellent programming language for writing TestPlans that range from extremely small and simple tests to extremely large and complex tests. The true power of DSBasic is realized when your test procedures change. The TestPlan program can be changed as needed in order to modify the TestPlan to meet your new requirements.

Cell Assistant is based on the concept of virtual channels. One of the first steps when configuring your system is to define and create all virtual channels. Each virtual channel has a type (e.g. Analog, Digital, Text, etc …) and a direction (input or output). Once you select the type and direction, you must "bind" a virtual channel to one of the physical channels that are provided by one of the hardware devices that are currently installed in the system. Once you select the device, the device driver will present a list of all channels that match the type and direction of the virtual channel that you are trying to create. Choose a physical channel to complete the creation of a new virtual channel. You now have

a virtual channel that is “bound” to a specific physical channel of a specific device. What does this mean? Read on.

All virtual channels are available to a running DSBasic TestPlan in the form of programming objects. When a TestPlan reads the value of a virtual channel object, it is accessing the last sampled value of that physical channel of that device (e.g. a voltage input, a temperature input, etc …). Similarly, when a running DSBasic TestPlan writes a value to a virtual channel object, the written data value goes to the underlying Cell Assistant hardware device driver and causes the appropriate command to be send to the device. As you can see, extremely sophisticated data acquisition and control system programs can be created using the DSBasic programming language.

Cell Assistant also allows the creation of various other types of virtual channels that are NOT connected to physical devices. These include counter channels, timer channels, channels that average other channels, and channels that are computed from one or more other channels (i.e. Calculated channels).

The data acquisition subsystem also performs limit checking and data logging functions. One or more limits can be placed on any channel. When a limit trips, Cell Assistant will automatically suspend execution of the currently running TestPlan and execute a special limit handling subroutine that is also written in DSBasic. Each limit is available to a running TestPlan in the form of a DSBasic object; thus, a running TestPlan can enable the limit, disable the limit, and modify limit setpoint values as it executes. Data file formats can be configured as needed. Again, they are under complete control of the DSBasic TestPlan in the form of programming objects.

Finally, the Cell Assistant software set also includes a set of custom ActiveX Controls that can be used to create high quality virtual instrument panels (VIP’s). One of these controls provides a rich set of user-interface controls such as analog meters, digital meters, strip charts, buttons, and pilot lights. Using Microsoft Visual Basic as a VIP builder, these controls can be used to build high quality user-interfaces to Cell Assistant. Absolutely no programming is required. Each control is designed to work exclusively with Cell Assistant. The user can modify the appearance of the control and which data channel it is connected to by merely changing the control properties. Another OLE control is also provided that provides access to Cell Assistant channel data but has no user-interface. This allows VIP’s to be created using virtual instrument ActiveX control sets from other vendors. These other control packages, on the other hand, do require several lines of Visual Basic code to be written per control; the user has to write code to move Cell Assistant channel data to and from these controls. Clearly, the best choice is to use the ActiveX controls package from DyneSystems. In addition to requiring no additional programming, they also provide many additional features that other control packages do not provide.

This overview is intended to give you a basic understanding of how Cell Assistant is organized. Each aspect mentioned above is described in greater detail in the chapters that follow.

Chapter 2 - Installation

Operating System Requirements

Cell Assistant for Windows™ runs under Microsoft Windows NT version 3.51 or 4.0. At the present time, Cell Assistant has not been tested on Windows 95, therefore, we do not support its use on Windows 95.

Hardware Requirements

The software is designed to run on personal computers running the above operating systems. The software could theoretically run on a different type of computer but this has not been verified. Microsoft does maintain a list of WIN32 compatible computers that have been verified with the Windows NT operating system and DyneSystemsCo. cannot guarantee the performance of the software if it is used on computers not on the list.

Computer Setup

Before the Cell Assistant software can be installed, the entire computer needs to be properly set up and configured. A procedure to do this is described in Appendix C. Even if the computer appears to already be configured, Appendix C should still be used as a check list to verify correct computer configuration. Appendix C describes how to install the Windows NT operation system (if necessary), how to make the correct settings to various NT applications, services, and drivers, how to install hardware devices (e.g. serial communications cards) and their respective drivers, how to install other support applications, etc …

Installation of Cell Assistant Software

The installation disks should have been provided with your system. It is very likely that Cell Assistant has already been loaded prior to ship time, however, you may still need these disks to re-install the software.

To run the installation program, insert the disk labeled “Cell Assistant for Windows™ Disk 1 of __”.

Windows NT Version 4.0:

Using the Windows NT Explorer tool, click on the 3½ Floppy icon to display all of the files contained in Disk 1. Double-click on the file **Setup.exe**. This will launch the Cell Assistant for Windows™ installation program.

Windows NT Version 3.51:

In program manager, select the ***File*** menu item, then the ***Run*** menu item. Type the following into the Command Line box: “A:\SETUP.EXE”. Please note that this assumes that the installation disk is inserted into drive A. If not, substitute the appropriate drive letter. Click the ***OK*** button.

The first screen is the welcome screen. Please read it and then click the ***Next*** button.

The next screen allows you to specify the directory in which the files will be installed. We recommend that you choose the CA4W subdirectory. Then click the ***Next*** button. The installation of

"Cell Assistant for Windows™" will now begin. You will be prompted to insert additional diskettes when they are needed.

Installation of Copy Protect Block

Cell Assistant requires a properly configured copy protect block to be installed on the parallel port of the computer. There are three levels of functionality available including Full, Edit Only, and Evaluation.

A copy-protect block configured for Full functionality will allow all features to function.

A copy-protect block configured for Edit Only functionality will only allow the editing and saving of TestSystems and TestPlans. The About box will show "Edit Only Version" when this type of copy protection block is installed.

A copy protect block configured for Evaluation functionality will allow all features to function but only for a preset number of times. After the system has been run that many times, it will no longer function. The About box will show "Evaluation Version" when this type of copy protection block is installed.

Chapter 3 - Getting Started

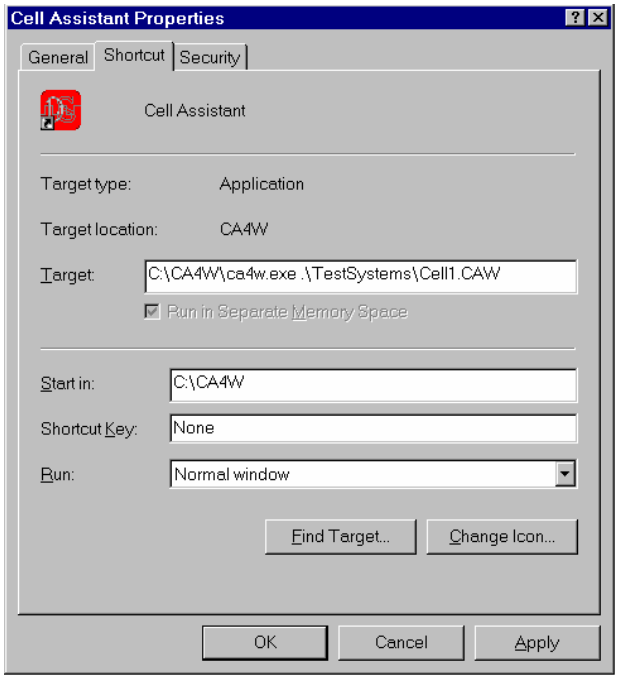
To launch Cell Assistant, double click the Cell Assistant icon in the Cell Assistant program group.

TestSystems

In order to proceed, you will need to open a TestSystem file. The TestSystem file is where all of the information about the system is kept. This information includes what hardware is configured, limits, channel configurations, etc. TestSystem files will have a .CAW file extension. Unless the hardware setup of a cell changes, only one TestSystem is usually needed for a cell.

To open a TestSystem file, click ***File*** on the menu bar at the top of the screen. If you are defining a new TestSystem from scratch, click ***New***. This brings up the TestSystem property sheet (the sections of the property sheet indicated by the tabs are called "property pages") that is used to configure the entire TestSystem. Since most systems come with at least a preliminary configuration done by Dyne Systems, click ***Open*** and select the file from the list or select one of the more recently used TestSystems at the bottom of the menu. This is where the frequently used files are kept. A single click on the desired file will open it.

Because only one TestSystem is typically used in a cell, Cell Assistant is able to automatically open a TestSystem when Cell Assistant is started. To do this, the name of the TestSystem file must be specified in the properties of the shortcut. A shortcut to Cell Assistant should be already created (if not, refer to Microsoft's documentation to create one). Click once on the Cell Assistant shortcut with the right mouse button. Click on the ***Properties*** menu item. Click on the ***Shortcut*** property page. Type the path of the TestSystem file in the Target text box as shown below.



While configuring a system it is usually wise to save your work periodically. How often? That depends on how much work you are willing to lose in case of an unforeseen event. To save your work, you can either click the ***Save*** item in the ***File*** drop-down menu or click the save button on the Tool Bar.

The following chapters describe in detail how to configure the TestSystem.

Password Protection

Password Levels

Cell Assistant has three levels of access, two of which are password protected.

The most basic level is the User level. This is the level accessed when Cell Assistant is started and does not require a password. This level allows the user to run a test (called a TestPlan) and to view various screens, but nothing can be changed.

The next level is the Calibration level. This level allows all functions of the user level plus the ability to calibrate the system. Cell Assistant uses “cal” as the default password for this level.

The highest level is the Configuration level. This level allows all functions of the user and calibration levels plus the ability to change anything in the system. Cell Assistant uses “cfg” as the default password for this level.

The current level is shown on the status bar (bottom of the window).

Entering Passwords

Click the padlock button on the Tool Bar or click the ***File*** menu and then the ***Password*** item. Enter the correct password. The level of access will be determined by the password that is entered. To log out, repeat the above steps and click the ***Logout*** button.

Changing Passwords

Log out if the present level is Calibration or Configuration. Click the padlock button on the Tool Bar or click the ***File*** menu and then the ***Password*** item. Then click the ***Change*** button. Type the old password that you wish to change in the ***Old Password*** dialog box. Then type the desired new password in the ***New Password*** dialog box. You must then retype the password in the ***Verify New Password*** dialog box to confirm that the correct password is being remembered by the system.

Main Menu Bar

The main menu bar contains the following items which will be described below.

File Main Menu Item

New Menu Item

Creates a new TestSystem file. The extension for a TestSystem file is ".CAW". This can also be accomplished by clicking the new button on the Tool Bar.

Open Menu Item

Allows the user to open a TestSystem file. This can also be accomplished by clicking the open button on the Tool Bar. A standard File Open dialog box will then be displayed. It will list all of the TestSystems that reside in the TestSystems subdirectory.

Close Menu Item

Allows the user to close a TestSystem file.

Save Menu Item

Saves a TestSystem file. This can also be accomplished by clicking the save button on the Tool Bar. If this is the first time the TestSystem is being saved, a standard File Save dialog box will be displayed which allows the operator to specify the name of the new TestSystem and the location where it should be saved. The default location is in the TestSystems subdirectory.

Save As Menu Item

Saves a TestSystem but gives the opportunity for the user to specify a different file name. A standard File Save As dialog box will be displayed.

Print Menu Item

Allows the user to print the TestSystem or TestPlan. Selecting this menu option (or clicking the print button on the Tool Bar) while the TestPlan edit/debug window is active will cause the TestPlan to be printed. If the TestPlan edit/debug window is not active, this will cause the TestSystem file to be printed.

Print Preview Menu Item

Allows the user to view the TestPlan or TestSystem printout before actually printing it.

Print Setup Menu Item

Allows the user to setup the printer.

Preferences Menu Item

This allows the user to configure various aspects of Cell Assistant. Setup of preferences will be described in the following section.

Password Menu Item

Allows user to Log In by providing the proper password or to Log out. This can also be done using the password button on the Tool Bar.

Recently Used File Area

This shows the last used files in the order that they were last used, with the top file being the last file used. The number of files displayed is determined in Preferences.

Exit

This causes Cell Assistant to close.

Setting System Preferences

General Property Page

TestSystem File Sharing

When Cell Assistant is running with an installed TestSystem, it prevents that TestSystem file from being written to by any other application. This could occur if a TestSystem file is stored on a different computer from the one that is running Cell Assistant and the file is accessed over a network. The first computer to open that file will have the ability to edit the file and save the new version.

When using a network, it is often desirable to allow more than one computer to read a TestSystem file. If this is the case, select the ***Grant READ access to other applications*** option button. Selecting the ***No access by other applications*** option button will allow only one computer to load a TestSystem file at a time. Any other computers that try to open that file will be denied access.

In addition, the numbers of files shown in the Most Recently Used File List section of the ***File*** menu can be set from this page.

Size of Most Recently Used File List

This is used to determine how many of the last used files will be displayed.

Network VIP's Data Service

Each Cell Assistant computer can make all of the data from the currently running TestSystem available to Virtual Instrument Panels (VIP;s) that are running on a network. This feature is enabled by checking the ***Enable connections by network VIP's*** checkbox. When this box is checked, Cell Assistant will start the "CA Data Server" which is a Windows NT service that runs in the background whenever Cell Assistant is running.

Network VIP's communicate with the CA Data Server service using TCP/IP. The ***TCP/IP Port*** that is entered will be the "well known" port number that all remote VIP's will try to connect to. In almost all cases, use the specified default value. In rare instances, your network administrator may need to change this value if this port is already used by another service that is running on your network. Keep in mind that if the port number is changed, all remote VIP's will also need to be updated.

E-Mail

The DSBasic language has been extended to include commands that allow a running TestPlan to send E-Mail. This feature must first be enabled by checking the ***TestPlans can send E-Mail*** checkbox. Cell Assistant will then initialize the E-Mail subsystem every time Cell Assistant is started. If E-Mail functionality is not needed and/or E-Mail capability is not available on the Cell Assistant computer, this checkbox should remain unchecked. The E-Mail feature is described in greater detail in other sections

View Interface Property Page

Interface Font Selection

This allows selection of the fonts used for editing and printing of a TestPlan and printing of a TestSystem. Any available font can be selected for the printing of a TestSystem; however, the font selection for editing and printing of a TestPlan is restricted to fixed-width fonts.

TestPlan Editor

Checking the ***Enable Auto-indent*** check box causes the TestPlan editor to automatically insert TAB characters into the TestPlan file (as needed) as you are writing DSBasic code. Whenever you press the **Enter** key, the next line will automatically "tab over" to the same indent level of the previous line.

View Options

Checking the ***Tool Bar Enabled*** check box will display the Tool Bar at the top of the Cell Assistant main window (just below the menu items). Checking the ***Status Bar Enabled*** check box will display the Status Bar at the very bottom of the Cell Assistant main window.

The Background Bitmap list box allows selection of a bitmap image to be used as the background in the Cell Assistant work space. Selecting the ***Tile*** option button will cause the image to be repeated in an array across the entire workspace. Selecting the ***Center*** option button will display the image in the center of the workspace.

Test Application Property Page

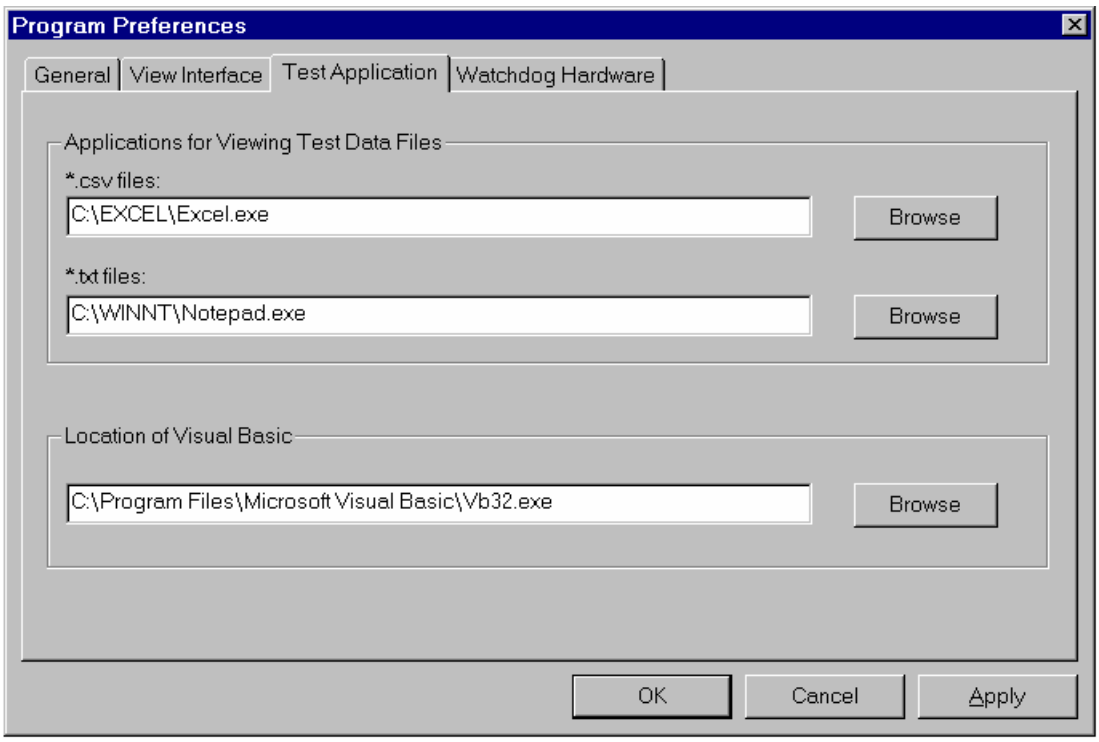
Applications for Viewing Test Data Files

One of the basic jobs of Cell Assistant is to create a data file while a test is running. This file can be created in two different formats - a text format (*.txt file) or an ASCII comma separated variable format (*.csv file). Once a data file has been saved, it often needs to be viewed. Viewing of a data file is done by opening that file with another application. Typically Microsoft Excel is used to view *.csv files and Microsoft Notepad is used to view *.txt files, but others can be used as well. Cell Assistant also comes with a **CSV File Viewer** application (CsvFileView.exe) that can be used to view the contents of *.csv files. It is a read-only application that only allows viewing of the file data; on the other hand, it is faster than Microsoft Excel and is capable of viewing much larger files than Excel. CsvFileView.exe is located in the main Cell Assistant subdirectory. The list boxes in this group box tell Cell Assistant the location of the application used to view the two types of data files when an operator selects ***View*** and then ***Data Files*** from the main menu.

Location of Visual Basic

Cell Assistant uses Microsoft Visual Basic to create Virtual Instrument Panels (VIP's). The list box in this group box tells Cell Assistant the location of Visual Basic.

The following shows a typical setup:



Watchdog Hardware Property Page

Watchdog Theory of Operation

The watchdog feature in Cell Assistant provides a method of shutting the system down in the event of a computer failure (hardware or software). To accomplish this, most systems include a small external watchdog timer module which is connected to a serial port. When Cell Assistant is running properly, the DTR bit is toggled approximately once each second. Each time the watchdog timer module senses the DTR bit transition from low to high it resets its internal timer. If this internal timer is not reset approximately every two seconds, the timer will expire and cause the output contacts of the module to open. These contacts should be wired up to the system such that anytime these contacts open the system shuts down.

If the watchdog circuit is properly installed, the system cannot be run unless the computer is on and Cell Assistant is running. In many cases it is desirable to be able to run without the computer on or without Cell Assistant running. This can be accomplished by using a switch to short the output contacts of the watchdog module, effectively bypassing the watchdog circuit. If this is done, however, there is a danger that a test can be inadvertently run with the watchdog bypassed. This will result in inadequate protection of the equipment. To eliminate this, an interlocking contact on the bypass switch (closed when the bypass switch contact is open and open when the bypass switch contact is closed) should be wired from the RTS line to the CTS line of the serial port. Cell Assistant will not allow the system to run a test if the contact from the RTS line to the CTS line of the serial port is open, which indicated that the watchdog is bypassed.

Serial Port Connection

This drop-down list box tells Cell Assistant which serial port the watchdog timer module is connected to. If there is no watchdog timer module, “Disabled” should be selected.

Watchdog Hardware Trips On

Not all errors in Cell Assistant need to cause the watchdog to expire. Some can be handled in the TestPlan by writing the proper DSBasic code. This is the case with DSBasic TestPlan errors and Device Communication errors. Unchecking an item prevents Cell Assistant from allowing the watchdog to expire if this error occurs. CAUTION - if one or both of these are unchecked, the TestPlan author is responsible for protecting against these errors.

Log

Any changes made to this property page are logged in this section. The log entries cannot be deleted.

TestPlan access to Watchdog

In many situations, a running TestPlan may want to be informed when the watchdog is about to be tripped. In other situations, a TestPlan may want to force a trip of the watchdog hardware as part of a shutdown or error handling routine. All of this can be implemented via channels that are provided by a pseudo-device called the SYSTEM device (See Chapter 4 - Hardware Devices for a description of devices in general). The SYSTEM Device cannot be added or removed like other devices; it is a driver that is permanently embedded in Cell Assistant. It provides various channels that can be accessed by a running TestPlan. See Appendix A for a list of all available SYSTEM Device Channels.

Edit Main Menu Item

This allows text in a TestPlan to be cut, copied, and pasted in the usual Windows way. This menu also contains the Find, Find Next, Replace, etc ... items that are standard in most text editors.

View Main Menu Item

When the ***View*** menu item is clicked, a drop-down menu with four items is displayed. These are ***Data File***, ***Flight Recorder File, Channel Data*** and ***Limit Sets***.

Clicking the ***Data File*** or ***Flight Recorder File*** item will display a standard File Open dialog box and display all available files in the DataFiles or FlightRecorder subdirectory respectively. After a file is selected, Cell Assistant will launch the application specified in the Preferences section (usually the DyneSystemsCo. CSV File Viewer Utility) based on the file type that was selected and open the selected file. Clicking the ***Channel Data*** item will launch an application (CHView.exe) which displays live data from the system. Similarly, clicking the ***Limit Sets*** item will launch an application (LimitSetView.exe) that displays a grid containing all currently defined limit values for all currently defined limit sets.

All three viewer utilities mentioned above (CSV File Viewer, Channel Viewer, and Limit Set Viewer) are part of the Cell Assistant software package. All are located in the Cell Assistant main subdirectory. They are explained in greater detail in the Utilities chapter of this manual.

Control Main Menu Item

Clicking the ***Control*** item will drop down a menu which enables the user to start or stop data acquisition or a TestPlan.

Data Acquisition

Click on ***Start Data Acquisition*** to cause the Data Acquisition task to run. This is used often when you wish to debug a VIP but you don't want to run a TestPlan. Click on ***Stop Data Acquisition*** to stop the Data Acquisition Task.

TestPlan

Click on ***TestPlan Run*** to run the default TestPlan. In order to stop a TestPlan which is currently running, click the ***TestPlan Stop*** item. Please note that when this happens, the equipment will be left in the state it was in at the instant the TestPlan is stopped and limits will no longer be checked.. Be sure to bring any rotating equipment to a stop manually or press the hardware Emergency Stop button.

Window Main Menu Item

Cell Assistant can have the TestSystem property sheet and multiple TestPlans open in its workspace at the same time. Clicking on ***Cascade*** will order the displayed TestPlans in the usual Windows way. If the TestPlans have been minimized to icons, clicking ***Arrange Icons*** will arrange the icons in the usual Windows fashion.

Help Main Menu Item

Cell Assistant features on-line help. Clicking the ***Index*** item on the drop-down menu will display a dialog box which allows the user to access a help index for the User's Manual, DSBasic Language Reference, Dialog Editor, and TestPlan Editor. Clicking on ***Using Help*** provides access to help for the help system itself. Clicking the ***About CA4W*** item or the About button on the Tool Bar will display a dialog box with version and copyright information.

Another way to get help information on a particular item is to click on the Help button on the Tool Bar. This will put a question mark next to the cursor. You can then click on any item that you need help on and the help information for that item will be displayed (if available).

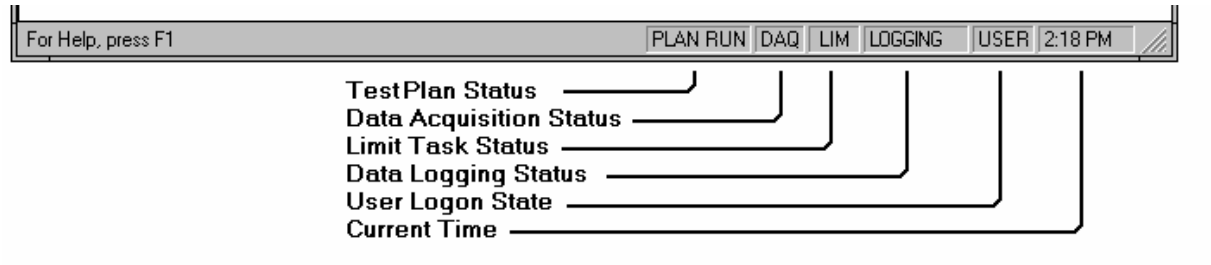
Tool Bar

The Tool Bar is located just below the main menu. It is shown below.

The buttons on the Tool Bar provide a faster way to access many of the frequently used menu items. Each has already been described in a previous section.

Status Bar

The Cell Assistant status bar is located at the bottom of the main window. It is shown below.



TestPlan Status
This indicator displays **PLAN RUN** whenever a TestPlan is running. If a TestPlan has been terminated by a limit exception handler, this indicator will display **ABORTED**. It is blank otherwise.

Data Acquisition Status
This indicator displays **DAQ** when data acquisition is running; otherwise, it is blank.

Limit Task Status
This indicator displays **LIM** when a limit task is running. If a limit exception is detected, it displays **TRIP**. The **LIM** status will replace the **TRIP** status when all limits return to normal. If the limit task is not running, this indicator is blank.

Data Logging Status
This indicator displays **LOGGING** when data logging is running. If no automatic logging has been started, this indicator displays **LOG OPEN** if any log file is currently open; otherwise, it is blank.

User Logon State
Displays the current user password level: **USER**, **CAL**, or **CFG**.

Current Time
Displays the current time of day.

Chapter 4 - Hardware Devices

In order to configure your hardware devices click the ***Hardware Devices*** tab on the TestSystem property sheet.

Adding a Hardware Device

The ***Devices*** box shows the hardware devices that are already installed and that CA4W has access to. To add one that's not on the list, click the ***Add*** button (you must be logged in at the configuration level). This will display a list of drivers (in the ***Add Device*** dialog box) that are available on your system.

Click on the desired driver which will cause it to become highlighted. Click the ***OK*** button. This will close the ***Add Device*** dialog box and add the highlighted driver to the ***Devices*** list.

Removing a Hardware Device

Clicking the ***Remove*** button will cause the highlighted Hardware Device to be removed from the list (you must be logged in at the configuration level). NOTE: The Device driver is still installed on your computer such that this Hardware Device can be added back to this TestSystem at a later date.

Hardware Device Setup

It is now necessary to configure the hardware device. Click on the hardware device to highlight it. Then click the ***Setup*** button (you must be logged in at the configuration level). This will bring up the property sheet required to configure the hardware/driver. Following are instructions for setting up the individual hardware devices.

Allen Bradley SLC 5/03 PLC setup

Cell Assistant communicates with an Allen Bradley SLC 5/03 PLC via a serial port. The driver can access data in Data File # 9 and Data File #10 in the PLC. Word and bit data from these files must be integrated into a Ladder Logic program in the PLC in order for Cell Assistant and the PLC to work together.

Port

Begin by clicking on the ***Port*** tab. Choose the baud rate from the ***Baud Rate:*** list by highlighting the desired baud rate. Select the desired ***Parity***, ***Data Bits***, ***Stop Bits***, and ***Flow Control*** from the option button group boxes. Select the COM port from the ***Connection*** drop-down list box.

Typical configuration is shown below:

Allen Bradley SLC 5/03 PLC Port setup	
Baud Rate	19200
Parity	None
Data Bits	8
Stop Bits	1
Flow Control	None

Exit by clicking the ***OK*** button to save your work. If you click the ***Cancel*** button, your settings will be forgotten.

Settings

The ***Station Number*** should be set to match the station number of the PLC that is being controlled. This is typically 1.

The ***Sample Frequency*** text box specifies the number of times per second the PLC will be sampled for data.

The next section of the Settings property page is the ***Data File 9 Allocation*** group box. This is where the number of digital input and analog input channels are set. Each channel maps to a unique 16-bit word and/or bit in Data File #9 in the PLC. The ***PLC Address Ranges*** group box displays the range of PLC addresses that are mapped to each group of channels.

All input channel names reference word/bit locations in the PLC. Cell Assistant can only read data from Data File # 9 in the PLC. This file is a user-defined file that must be created by the PLC programmer (file type = Binary). The size of the file must be large enough to accommodate all input channels. Digital Input channels are mapped to individual bits in the PLC and, thus, a 16-bit word in Data File # 9 can contain 16 digital input bits that can be read by Cell Assistant. Analog Input channels, on the other hand, are each mapped to an entire 16-bit word in Data File #9.

When creating Data File # 9 in the PLC, set the size as follows:

No. of Elements = (Digital Inputs / 16) + (Analog Inputs)

Channel data is stored in the following order in PLC Data File #9:

1) Digital Inputs Stored at beginning of file.
2) Analog Inputs Stored immediately after Digital Inputs.

The next section of the Settings property page is the ***Data File 10 Allocation*** group box. This is where the number of digital output and analog output channels are set. Each channel maps to a unique 16-bit word in Data File #10 in the PLC. The ***PLC Address Ranges*** groupbox displays the range of PLC addresses that are mapped to each group of channels.

All output channel names reference word/bit locations in the PLC. Cell Assistant can only write to data in Data File # 10 in the PLC. This file is a user-defined file that must be created by the PLC programmer (file type = Binary). The size of the file must be large enough to accommodate all output channels. Digital Output channels are mapped to the least significant bit of a 16-bit word in Data File #10. This is somewhat inefficient but is necessary because the SLC 5/03 CPU does not support any commands that allow Cell Assistant to modify an individual bit in a 16-bit word. Analog Output channels, on the other hand, are each mapped to an entire 16-bit word in Data File #10.

When creating Data File # 10 in the PLC, set the size as follows:

No. of Elements = (Digital Outputs) + (Analog Outputs)

Channel data is stored in the following order in PLC Data File #10:

1) Digital Outputs Stored at beginning of file.
2) Analog Outputs Stored immediately after Digital Outputs.

The resulting names of all channels match the naming conventions used by RSLogix 500 ™, the standard programming software package used to program SLC 500 series PLC's. Thus, the name of each channel clearly indicates which word/bit it is mapped to in the PLC.

NOTE: It is extremely important to make initial allocations that are larger than the initial design would normally dictate. In fact, the initiall allocations should anticipate all future needs. Since all channel names are created based on these allocations, it is important that the allocation never be changed. If they are, a severe amount of PLC and CA4W TestPlan reprogramming may be required.

Exit by clicking the ***OK*** button to save your work. If you click the ***Cancel*** button, your settings will be forgotten.

Available Channels

Communication to the Allen Bradley SLC 5/03 PLC is accomplished through channels. Once the PLC driver is installed and configured, the necessary channels are made available to the system. Please see Appendix A for detailed channel availability.

ASCII File setup

The ASCII File driver is used to read data that exists in a simple ASCII text file. This data in this file is typically saved there by another application. Each row in this file contains one piece of data.

NOTE: The ASCII file should reside on the local computer that is running the Cell Assistant software. While the driver is capable of accessing files on other computers via a network connection, the real time efficiency of the Cell Assistant software can be compromised by network delays and network timeouts.

Settings

The ***ASCII File*** is the name of the file that is periodically opened, accessed, and closed by this device driver. If you know the name and location of this file, enter the complete path and filename; otherwise, click the ***Browse*** button to locate the data file that contains the data to be sampled.

The ***No. of channels in file*** specifies the number of lines of data to be read by the driver per file access. You can specify more or less than the actual number of lines of data in the file; however, the driver will create an analog input device channel for exactly the number of channels specified in this box. If the file actually contains more lines of data, the additional lines of data will not be read. If the file actually contains less lines of data, the extra channels that have been created will be set to 0.

The ***Sampling Interval*** is the number of seconds (or minutes) between successive file accesses. Click the down arrow to the right of the ***Sampling Interval*** drop-down text box. Select the desired sampling interval from the list.

The settings page also contains a check box that can allow the driver to keep going if file access errors occur. Normally, all file access errors (e.g. file not found, read errors, etc ...) are treated as fatal errors. An error dialog box is displayed and the watchdog hardware will trip. In some situations, however, it is known that "file not found" errors are expected to occur during normal operation. Checking this box tells the driver to ignore the error and try again later. The watchdog hardware will continue to be updated as if the error never occurred.

Exit by clicking the ***OK*** button to save your work. If you click the ***Cancel*** button, your settings will be forgotten.

Diagnose

For diagnostic purposes, a simple control screen has been provided. Once the system is set up, the ***Diagnose*** property pages can be used to check operation of the system. Clicking the ***Data*** button updates the screens.

Available Channels

Access to the ASCII File is accomplished through channels. Once the driver is installed and configured, the necessary channels are made available to the system. Please see Appendix A for detailed channel availability.

AVL Fuel Balance setup

Port

Begin by clicking on the ***Port*** tab. Choose the baud rate from the ***Baud Rate:*** list by highlighting the desired baud rate. Select the desired ***Parity***, ***Data Bits***, ***Stop Bits***, and ***Flow Control*** from the option button group boxes. Select the COM port from the ***Connection*** drop-down list box.

Typical configuration is shown below:

AVL Fuel Balance Port setup	
Baud Rate	19,200
Parity	None
Data Bits	8
Stop Bits	1
Flow Control	None

Exit by clicking the ***OK*** button to save your work. If you click the ***Cancel*** button, your settings will be forgotten.

Diagnose

For diagnostic purposes, a simple control screen has been provided. Once the system is set up, the ***Diagnose*** property pages can be used to check operation of the system. Clicking the ***Data*** button updates the screens.

In addition, a simple terminal emulator has been included to allow the user to send any desired data to the device and observe the response (if any). To do this, type the desired command in the ***Send:*** text

box and click the ***Send*** button. Any response will be displayed next to the ***Rx:*** text. Several common ***System*** and ***Measurement*** commands are available as buttons on this page. The button names correspond to channel names which are listed in Appendix A.

Available Channels

Communication to the AVL Fuel Balance is accomplished through channels. Once the driver is installed and configured, the necessary channels are made available to the system. Please see Appendix A for detailed channel availability.

Baldor 22H setup

Port

Begin by clicking on the ***Port*** tab. Choose the baud rate from the ***Baud Rate:*** list by highlighting the desired baud rate. Select the desired ***Parity***, ***Data Bits***, ***Stop Bits***, and ***Flow Control*** from the option button group boxes. Select the COM port from the ***Connection*** drop-down list box.

Typical configuration is shown below:

Baldor Series 22H Port setup	
Baud Rate	19,200
Parity	None
Data Bits	8
Stop Bits	1
Flow Control	None

Exit by clicking the ***OK*** button to save your work. If you click the ***Cancel*** button, your settings will be forgotten.

Settings

Next click the ***Settings*** tab. There are two items on the Settings property page. The first item is the ***Digital Inputs*** group box which contains four dropdown listboxes. Four of the digital input values that this device driver continuously samples are programmable by the user. Using terminology from the "Baldor Series 22H Line Regen AC Flux Vector Control Installation and Operators Manual", these digital inputs are Opto Output #1 (Parameter 1501), Opto Output #2 (Parameter 1502), Opto Output #3 (Parameter 1503), and Opto Output #4 (Parameter 1504). Each can be programmed to read back any one of approximately 15 values (see Operators Manual). This selection should be done using the keypad and display on the vector drive control panel. The selections that are made in the control should also be made on this page. Cell Assistant uses these settings to display the correct channel labels on the ***Diagnostic*** page.

NOTE: Even though each of the four channels can be configured to read back any one of approximately 15 channels, Cell Assistant requires that Opto Output #4 (P1504) is configured to read back the "Fault" signal.

The second item is the ***Analog Inputs*** group box which contains two dropdown listboxes. Two of the analog input values that this device driver continuously samples are programmable by the user. Using terminology from the "Baldor Series 22H Line Regen AC Flux Vector Control Installation and Operators Manual", these analog inputs are Analog Output #1 (Parameter 1508) and Analog Output #2

(Parameter 1509). Each can be programmed to read back any one of approximately 23 values (see Operators Manual). This selection should be done using the keypad and display on the vector drive control panel. The selections that are made in the control should also be made on this page. Cell Assistant uses these settings to display the correct channel labels in the ***Diagnostic*** page.

Exit by clicking the ***OK*** button to save your work. If you click the ***Cancel*** button, your settings will be forgotten.

Diagnose

For diagnostic purposes, a simple control screen has been provided. Once the system is set up, the ***Diagnose*** property page can be used to check operation of the system. Clicking the ***Data*** button on this page causes one sample iteration to occur. The sampled data values are displayed in the ***Data*** group boxes.

A simple terminal emulator has been included on this diagnose page to allow the user to send any desired command to this device and observe the response (if any). To do this, type the desired command in the ***Send:*** text box and click the ***Send*** button. The operator must know which commands will return a response and which do not. If a response is expected for a specific command, the ***Send with Reply*** checkbox should be checked prior to clicking the ***Send*** button. Any response will be displayed next to the ***Rx:*** text.

This page also contains two button that can be used to quickly ***Enable*** or ***Stop*** the drive. Clicking either button will send the appropriate command to the drive control.

Available Channels

Communication to the Baldor Series 22H Vector Drive is accomplished through channels. Once the Baldor Series 22H driver is installed and configured, the necessary channels are made available to the system. Please see Appendix A for detailed channel availability.

Companion setup

Port

Begin by clicking on the ***Port*** tab. Choose the baud rate from the ***Baud Rate:*** list by highlighting the desired baud rate. Select the desired ***Parity***, ***Data Bits***, ***Stop Bits***, and ***Flow Control*** from the option button group boxes. Select the COM port from the ***Connection*** drop-down list box.

Typical configuration is shown below:

Companion Port setup	
Baud Rate	115,200
Parity	None
Data Bits	8
Stop Bits	1
Flow Control	None

Exit by clicking the ***OK*** button to save your work. If you click the ***Cancel*** button, your settings will be forgotten.

Settings

At the present time, there are no settings that can be changed by the operator. In the future, other combinations of backplanes and sampling rates will be available. At this time, the ***Settings*** property page displays the one (and only) configuration that is supported.

Configure

The ***Configure*** property page is used to define the 5B modules that are installed in each slot of each backplane. Click the appropriate radio button (***1*** through ***7***) to view a graphical representation of the selected backplane. Then double-click on the appropriate slot to specify (or respecify) the 5B module that is currently installed in the selected slot. A dialog box will be displayed that lists all currently supported 5B modules. Select the 5B module for this slot and click the ***OK*** button.

Diagnose

For diagnostic purposes, two simple control screens have been provided: one to diagnose Analog Input modules and one to diagnose Analog Output modules. Once the system is set up, the ***Diagnose AI*** and ***Diagnose AO*** property pages can be used to check operation of the system.

Available Channels

Communication to the Companion is accomplished through channels. Once the driver is installed and configured, the necessary channels are made available to the system. Please see Appendix A for detailed channel availability.

Daytronic System 10 setup

Port

Begin by clicking on the ***Port*** tab. Choose the baud rate from the ***Baud Rate:*** list by highlighting the desired baud rate. Select the desired ***Parity***, ***Data Bits***, ***Stop Bits***, and ***Flow Control*** from the option button group boxes. Select the COM port from the ***Connection*** drop-down list box.

Typical configuration is shown below:

Daytronic System 10 Port setup	
Baud Rate	115,200
Parity	None
Data Bits	8
Stop Bits	1
Flow Control	DTR/CTS

Exit by clicking the ***OK*** button to save your work. If you click the ***Cancel*** button, your settings will be forgotten.

Settings

The sample frequency is the number of times per second the System 10 will be polled for data. Click the down arrow to the right of the ***Sample Frequency*** drop-down text box. Select the desired sample frequency from the list.

The ***DownLoad Delay:*** is the number of milliseconds that the system will wait between the sending of each character to the System 10. This is typically set to 50.

The Reset Delay is the number of milliseconds that the system will wait after sending the RST command to the Daytronic System 10. This is typically set to 500.

Click the ***DownLoad Prompts:*** check box if you wish to be prompted during the download.

Exit by clicking the ***OK*** button to save your work. If you click the ***Cancel*** button, your settings will be forgotten.

Configure

The main feature of this property page is the ***Slot/Card*** list box. This box has 20 rows - one for each slot in a Daytronic System 10 twenty card rack. If you have a rack with less than 20 slots, use only the number you have (starting at one). Begin configuration by selecting the slot that you wish to install a card in and click the ***Add*** button. The ***Add Card*** dialog box shows a list of the available cards. Choose the card from the list that you wish to add and click the ***OK*** button. The name of that card will now appear in the slot. If you wish to move the card (or any card), click the ***Move Up*** or ***Move Down*** button.

Describing every card is beyond the scope of this manual, however, some general points are covered below. Once the card is shown in the proper slot, it can be configured by clicking the ***Modify*** button. The ***General Channel Setup*** dialog box contains a list box labeled ***Channels***. This box contains a row for every channel contained on the card. Choose the channel you wish to configure, then select a type from the ***Type*** drop-down list box. You can also type in a label in the ***Label*** edit box. This is not meant to be a channel name (that will be done later). If used, it should designate a hardware connection (such as “T/C Jack 1” or “Pressure 1” rather than “Oil Temperature” or “Oil Pressure”). The ***Slope***, ***Offset***, and ***Filter*** edit boxes allow entry of slope and offset (m and b in the conversion Y = mX + b or “EMM” and “BEE” in Daytronic Mnemonics) and Filter (FIL in Daytronic Mnemonics). These parameters configure the Daytronic System 10 itself so that the conversions are done in the System 10 rather than in the Cell Assistant computer. This helps reduce processing time.

After a channel has been set up, be sure to click the ***Accept*** button before choosing any other channel or your work will be forgotten.

Once all of the cards are properly configured, the information must be sent to the System 10 in order to configure it. To do this, click the ***Download*** button and follow the instructions. After the download is complete, Cell Assistant uploads the information from the Daytronic and fills in the boxes in the various dialogs. As each item is uploaded, it is compared with the value that was downloaded. All discrepancies are immediately displayed and should be reviewed by the operator. Most are harmless (e.g. 0.0 was downloaded but the System 10 returned a 0.000 when the same value was uploaded). This example can clearly be dismissed by the operator. Other discrepancies may dictate a reboot of the System 10 followed by a redownload of configuration by Cell Assistant.

Click ***OK*** to return to the TestSystem property sheet.

Diagnose

Once the system is configured, the ***Diagnose*** property page can be used to check operation. Note that the ***Data*** button gets data from the System 10 and updates the screen. The ***Data*** button retrieves and displays either Analog input data or Digital input data. Press the ***Analog Inputs*** or ***Digital Inputs*** radio button to select the data type to be displayed.

There is also a basic terminal emulator feature included here. In order to use it, the Daytronic System 10 must be in the ASCII communication mode. To do this, be sure the EEPROM switch is in the write position and type "ASN=0" in the ***Send*** text box. Then click the ***Send*** button. You can now type any command you want into the ***Send*** text box and send it to the Daytronic System 10. Responses to the command will show up next to the ***Rx:*** text. When you are finished, be sure to type "ASN=1" in the ***Send*** text box and click the ***Send*** button. This will return the Daytronic System 10 back to binary communications mode (required for Cell Assistant). It's also a good idea to take the EEPROM switch out of the write position.

Available Channels

Communication to the Daytronic System 10 is accomplished through channels. Once the driver is installed and configured, the necessary channels are made available to the system. Please see Appendix A for detailed channel availability.

Direct Logic 205 PLC setup

Cell Assistant communicates to a Direct Logic 205 PLC via a serial port. The driver can access Control Relays and some V Memory locations in the PLC. Control Relays are accessed via Digital Input and Output channels in Cell Assistant. Similarly, V Memory locations are accessed via Analog Input and Output channels in Cell Assistant. These Control Relays and V Memory locations must be integrated into a Ladder Logic program in the PLC in order for Cell Assistant and the PLC to work together. NOTE: This driver assumes that the Direct Logic 205 system is using a DL250 CPU module.

Port

Begin by clicking on the ***Port*** tab. Choose the baud rate from the ***Baud Rate:*** list by highlighting the desired baud rate. Select the desired ***Parity***, ***Data Bits***, ***Stop Bits***, and ***Flow Control*** from the option button group boxes. Select the COM port from the ***Connection*** drop-down list box.

Typical configuration is shown below:

Direct Logic 205 PLC Port setup	
Baud Rate	9600
Parity	Even
Data Bits	8
Stop Bits	1
Flow Control	None

Exit by clicking the ***OK*** button to save your work. If you click the ***Cancel*** button, your settings will be forgotten.

Settings

The first section of the Settings property page is the ***Control Relays*** group box. This is where the number of digital input and digital output channels are specified. Cell Assistant can only read from and force (write to) Control Relays in the PLC. The PLC programmer should reserve a contiguous block of Control Relays that are reserved for communications with Cell Assistant. This block of Control Relays will be partitioned into two contiguous groups. The first group will be written to by Cell Assistant via Digital Output channels. When these Control Relays are used in a ladder program,

only the relay contacts should be placed somewhere in a ladder rung (if you accidentally place the associated coil in your ladder program, the PLC will then control the state of the contacts and Cell Assistant will lose control). The second group of Control Relays will be read by Cell Assistant via Digital Input channels. When these Control Relays are used in a ladder program, only the coil portion of the relay is typically placed in a ladder rung.

The ***First Relay*** text box specifies the Control Relay number of the first relay to be connected to a Cell Assistant digital output channel. The value that is entered should be an octal value in order to remain consistent with the Direct Logic 205 User Manual. The number of digital output channels required is specified in the ***No. of Outputs*** text box. The number of digital input channels required is specified in the ***No. of Inputs*** text box. Both values should be entered as decimal values). Thus, the PLC programmer should reserve ***(No. of Outputs + No. of Inputs)*** Control Relays starting with ***First Relay*** in the PLC program.

It is important to keep in mind that Cell Assistant uses these settings to create device channels for the PLC driver and the names of these device channels include the relay number of each Control Relay. As a result, it's a good idea to try to anticipate the number of PLC input and output relays that will be needed and configure the driver for that amount and possibly more (just in case your PLC design needs to be modified). For example, if additional Control Relays connected to digital output channels are needed, they can easily be added by increasing the ***No. of Output***s setting; however, this will cause the numbering sequence (and hence the device channel names) for all of the digital input channels to change. This will require a significant amount of channel reconfiguration later on to compensate for these changes. The major point here is this: determine the number of digital input and digital output channels that are needed, add extra input and output channels to allow for future expansion, configure the driver based on these numbers, and (hopefully) never change the driver configuration again.

The default setting is for 50 output relays starting with relay 0 followed by 50 input relays. The PLC program should therefore reserve 100 Control Relays - C0 through C143 (octal values). Control Relays C0 through C61 (octal) will be contacts only (forced by Cell Assistant via digital output channels), and Control Relays C62 through C143 (octal) will be coils only (read by Cell Assistant via digital input channels).

The next section of the Settings property page is the ***Analog Inputs*** group box. Specify the number of analog input channels that are needed in the ***No. of Inputs*** text box. The default value is 0. Specify the starting V Memory address of the first analog input in the ***Start Address*** text box. The default value is 1400 (octal). All input channels are numbered consecutively starting with the specified ***Start Address*** value. It is the responsibility of the PLC programmer to get analog input data into these V Memory locations. The Direct Logic DL250 CPU scans all inputs for a given input module and stores the data in contiguous V Memory locations. The exact V memory address is programmable. If more than one input module is installed, the PLC programmer must guarantee that input channel data from all modules is stored contiguously.

All analog input data must be converted from the Direct Logic format to standard 2's complement format before it can be used in the PLC or Cell Assistant. If the PLC program processes the input data, it must perform this conversion anyway so it should store the converted values in the V Memory locations that are accessed by Cell Assistant. In this case the ***2's complement conversion done in PLC*** checkbox should be checked. If the PLC merely acquires the data and passes it directly to Cell Assistant, this checkbox should NOT be checked. In this case, the PLC program can store the raw input data directly into the V Memory locations that Cell Assistant accesses and then Cell Assistant will perform the conversion to 2's complement.

The next section of the Settings property page is the ***Analog Outputs*** group box. Specify the number of analog output channels that are needed in the ***No. of Outputs*** text box. The default value is 0. Specify the starting V Memory address of the first analog output in the ***Start Address*** text box. The default value is 1440 (octal). All output channels are numbered consecutively starting with the specified ***Start Address*** value. It is the responsibility of the PLC programmer to get analog output data from these V Memory locations and write them to the various output modules that are installed..

All analog output data must be converted to the Direct Logic format before it can be written to an output module. If the PLC program processes the output data prior to outputting it, it must perform this conversion anyway. So it should retrieve the 2's complement values from the V Memory locations that are accessed by Cell Assistant, process that data as needed, and then convert it to the Direct Logic format prior to writing the values to the output modules. In this case the ***2's complement conversion done in PLC*** checkbox should be checked. If the PLC merely passes the data directly from Cell Assistant to the output modules, this checkbox should NOT be checked. In this case, Cell Assistant will perform the conversion to 2's complement prior to storing the output data in V Memory..

The ***Sample Frequency*** text box specifies the number of times per second the PLC will be sampled for data.

The ***Slave Address*** should be set to match the slave address of the PLC that is being controlled. This is typically 1.

Exit by clicking the ***OK*** button to save your work. If you click the ***Cancel*** button, your settings will be forgotten.

Available Channels

Communication to the Direct Logic 205 PLC is accomplished through channels. Once the PLC driver is installed and configured, the necessary channels are made available to the system. Please see Appendix A for detailed channel availability.

DTC setup

A DTC Digital Throttle Control is typically connected to a Dyn-Loc. Cell Assistant communicates with the DTC via the Dyn-Loc/DTC driver. The DTC can also be connected directly to the Cell Assistant computer via a serial port. This driver is used in this configuration.

Port

Begin by clicking on the ***Port*** tab. Choose the baud rate from the ***Baud Rate:*** list by highlighting the desired baud rate. Select the desired ***Parity***, ***Data Bits***, ***Stop Bits***, and ***Flow Control*** from the option button group boxes. Select the COM port from the ***Connection*** drop-down list box.

Typical configuration is shown below:

DTC Port setup	
Baud Rate	19,200
Parity	None
Data Bits	8
Stop Bits	1
Flow Control	RTS/CTS

Exit by clicking the ***OK*** button to save your work. If you click the ***Cancel*** button, your settings will be forgotten.

Diagnose

For diagnostic purposes, a simple control screen has been provided. Once the system is set up, the ***Diagnose*** property pages can be used to check operation of the system. Clicking the ***Data*** button updates the screens.

In addition, a simple terminal emulator has been included to allow the user to send any desired data to the device and observe the response (if any). To do this, type the desired command in the ***Send:*** text box and click the ***Send*** button. Any response will be displayed next to the ***Rx:*** text. Several common ***System*** and ***Measurement*** commands are available as buttons on this page. The button names correspond to channel names which are listed in Appendix A.

Available Channels

Communication to the DTC is accomplished through channels. Once the driver is installed and configured, the necessary channels are made available to the system. Please see Appendix A for detailed channel availability.

Dyn-Loc/DTC setup

Cell Assistant communicates to the Dyn-Loc via a serial port. A DTC Digital Throttle Control, if used, is typically connected to the Dyn-Loc.

Port

Since communication to the DTC does not require another serial port, only the port to the Dyn-Loc needs to be configured. Begin by clicking on the ***Port*** tab. Choose the baud rate from the ***Baud Rate:*** list by highlighting the desired baud rate. Select the desired ***Parity***, ***Data Bits***, ***Stop Bits***, and ***Flow Control*** from the option button group boxes. Select the COM port from the ***Connection*** drop-down list box.

Typical configuration is shown below:

Dyn-Loc/DTC Port setup	
Baud Rate	19,200
Parity	None
Data Bits	8
Stop Bits	1
Flow Control	RTS/CTS

Exit by clicking the ***OK*** button to save your work. If you click the ***Cancel*** button, your settings will be forgotten.

Settings

Next click the ***Settings*** tab. There are three items on the Settings property page. The first is the ***Options*** group box which contains two check boxes. When the ***Binary Set Points*** check box is checked, the system sends set points to the Dyn-Loc over the serial port in binary format. This results in faster transmission. This will usually be checked unless an older version Dyn-Loc is used which doesn't support binary setpoints. The ***1/10 LAC*** check box should be checked only if that modification has been done to the Dyn-Loc.

The second item on the Settings property page is the ***Set Point Multipliers*** group box. Set Point multipliers are necessary because the Dyn-Loc can be configured for different numbers of decimal places in the displays. Even though these decimal places are there, the serial port commands must be given in integer (no decimal places) numbers, which is often difficult to remember when issuing commands. For example, if the torque display is set for two decimal places and the desired setpoint was 12.00 pound feet of torque, the number sent over the serial port would be 1200. This was often confusing, so Set Point Multipliers were added to eliminate the need to remember this.

If the display in question (speed or torque) has no decimal places (i.e. 1234), the multiplier should be set to 1. If it has one decimal place (i.e. 123.4) the multiplier should be set to 10. And if it has two decimal places (i.e. 12.34), it should be set to 100. Once these are set correctly, setpoints can be written including decimal points. For example (similar to the above example), if the torque display is set for two decimal places and the desired setpoint was 12.00 pound feet of torque, the number written in the TestPlan would be 12.00 or just 12.

The last item on the Settings property page is the ***Sample Frequency***. This determines the number of times per second the Dyn-Loc will be polled for data.

Exit by clicking the ***OK*** button to save your work. If you click the ***Cancel*** button, your settings will be forgotten.

Diagnose

For diagnostic purposes, two simple control screens have been provided. Once the system is set up, the ***Diagnose Dyn-Loc*** and ***Diagnose DTC*** property pages can be used to check operation of the system. Clicking the ***Data*** button on either of the pages updates the respective screens.

In addition, a simple terminal emulator has been included on each diagnose page to allow the user to send any desired data to the device and observe the response (if any). To do this, type the desired command in the ***Send:*** text box and click the ***Send*** button. Any response will be displayed next to the ***Rx:*** text.

Available Channels

Communication to the Dyn-Loc and the DTC is accomplished through channels. Once the Dyn-Loc/DTC driver is installed and configured, the necessary channels are made available to the system. Please see Appendix A for detailed channel availability.

Modicon PLC setup

Cell Assistant communicates to a Modicon PLC via a serial port. The software can only read coils (digital inputs) and force coils (digital outputs). In order to communicate, coils must be configured in the ladder logic program in the PLC.

Port

Begin by clicking on the ***Port*** tab. Choose the baud rate from the ***Baud Rate:*** list by highlighting the desired baud rate. Select the desired ***Parity***, ***Data Bits***, ***Stop Bits***, and ***Flow Control*** from the option button group boxes. Select the COM port from the ***Connection*** drop-down list box.

Typical configuration is shown below:

Modicon PLC Port setup	
Baud Rate	9600
Parity	Even
Data Bits	8
Stop Bits	1
Flow Control	None

Exit by clicking the ***OK*** button to save your work. If you click the ***Cancel*** button, your settings will be forgotten.

Settings

The first section of the Settings property page is the ***Digital I/O Coils (0xxxx)*** group box. This is where the number of digital input and digital output channels are specified. Cell Assistant can only read from and force (write to) coils in the PLC. Coils that are forced are considered input coils (with respect to the PLC) and are linked to Cell Assistant through digital output channels. Coils that are read are considered output coils (with respect to the PLC) and are linked to Cell Assistant through digital input channels. The number of digital output channels required is specified in the ***No. of Outputs*** text box. Clearly, this number depends on the number of PLC input coils (coils which will be written to by Cell Assistant via digital output channels) that are needed. The number of digital input channels required is specified in the ***No. of Inputs*** text box. This number depends on the number of PLC output coils (coils which will be read from by Cell Assistant via digital input channels) that are needed.

When writing PLC programs, coils with numbers matching the address numbers in Cell Assistant must be created. The address of the first input coil will be the address specified in the ***Starting Address*** text box and all other input coils will follow in sequential order. Following at the next sequential address will be the address of the first output coil. All other output coils will follow in sequential order.

It is important to keep in mind that Cell Assistant uses these settings to create device channels for the PLC driver and the names of these device channels include the PLC address of each coil. As a result, it's a good idea to try to anticipate the number of PLC input and output coils that will be needed and configure the driver for that amount and possibly more (just in case your PLC design needs to be modified). For example, if additional input coils (and hence more digital output channels) are needed, they can easily be added by increasing the ***No. of Output***s setting; however, this will cause the addressing sequence (and hence the device channel names) for all of the digital input channels to

change. This will require a significant amount of channel reconfiguration later to compensate for these changes. The major point here is this: determine the number of digital input and digital output channels that are needed, add extra input and output coils to allow for future expansion, configure the driver based on these numbers, and (hopefully) never change the driver configuration again.

The default setting is for a starting address of 0300, with 64 inputs and 64 outputs. The PLC program should therefore contain 128 coils with addresses of 00300 through 00427. Coils 300 through 363 will be input coils (forced by Cell Assistant via digital output channels), and coils 364 through 427 will be output coils (read by Cell Assistant via digital input channels).

The next section of the Settings property page is the ***Analog Input Registers (3xxxx)*** group box. Specify the number of analog input registers that are needed in the ***No. of Inputs*** text box. The default value is 0. Specify the starting address of the first analog input register in the ***Starting Address*** text box. The default value is 0000 (resulting in a PLC I/O Reference address of 30000). All input registers are numbered consecutively starting with the specified ***Starting Address*** value.

The next section of the Settings property page is the ***Analog Output Registers (4xxxx)*** group box. Specify the number of analog output registers that are needed in the ***No. of Outputs*** text box. The default value is 0. Specify the starting address of the first analog output register in the ***Starting Address*** text box. The default value is 0000 (resulting in a PLC I/O Reference address of 40000). All output registers are numbered consecutively starting with the specified ***Starting Address*** value.

The ***Sample Frequency*** text box specifies the number of times per second the PLC will be sampled for data.

The ***Slave Address*** should be set to match the slave address of the PLC that is being controlled. This is typically 1.

Exit by clicking the ***OK*** button to save your work. If you click the ***Cancel*** button, your settings will be forgotten.

Available Channels

Communication to the Modicon PLC is accomplished through channels. Once the PLC driver is installed and configured, the necessary channels are made available to the system. Please see Appendix A for detailed channel availability.

Pierburg Emissions Bench setup

Port

Begin by clicking on the ***Port*** tab. Choose the baud rate from the ***Baud Rate:*** list by highlighting the desired baud rate. Select the desired ***Parity***, ***Data Bits***, ***Stop Bits***, and ***Flow Control*** from the option button group boxes. Select the COM port from the ***Connection*** drop-down list box.

Typical configuration is shown below:

Pierburg Emissions Bench Port setup	
Baud Rate	9600
Parity	Even
Data Bits	7
Stop Bits	1
Flow Control	None

Exit by clicking the ***OK*** button to save your work. If you click the ***Cancel*** button, your settings will be forgotten.

Diagnose

For diagnostic purposes, a simple control screen has been provided. Once the system is set up, the ***Diagnose*** property pages can be used to check operation of the system. Clicking the ***Data*** button updates the screens.

In addition, a simple terminal emulator has been included to allow the user to send any desired data to the device and observe the response (if any). To do this, type the desired command in the ***Send:*** text box and click the ***Send*** button. Any response will be displayed next to the ***Rx:*** text. Several common ***Commands*** are available as buttons on this page. The button names correspond to channel names which are listed in Appendix A.

Available Channels

Communication to the Pierburg Emissions Bench is accomplished through channels. Once the driver is installed and configured, the necessary channels are made available to the system. Please see Appendix A for detailed channel availability.

Pierburg PDP CVS setup

Port

Begin by clicking on the ***Port*** tab. Choose the baud rate from the ***Baud Rate:*** list by highlighting the desired baud rate. Select the desired ***Parity***, ***Data Bits***, ***Stop Bits***, and ***Flow Control*** from the option button group boxes. Select the COM port from the ***Connection*** drop-down list box.

Typical configuration is shown below:

Pierburg PDP CVS Port setup	
Baud Rate	9600
Parity	Even
Data Bits	7
Stop Bits	2
Flow Control	None

Exit by clicking the ***OK*** button to save your work. If you click the ***Cancel*** button, your settings will be forgotten.

Diagnose

For diagnostic purposes, a simple terminal emulator has been included to allow the user to send any desired data to the device and observe the response (if any). To do this, type the desired command in the ***Send:*** text box and click the ***Send*** button. Any response will be displayed next to the ***Rx:*** text.

Available Channels

Communication to the Pierburg PDP CVS is accomplished through channels. Once the driver is installed and configured, the necessary channels are made available to the system. Please see Appendix A for detailed channel availability.

Sartorius LC3201D setup

Port

Begin by clicking on the ***Port*** tab. Choose the baud rate from the ***Baud Rate:*** list by highlighting the desired baud rate. Select the desired ***Parity***, ***Data Bits***, ***Stop Bits***, and ***Flow Control*** from the option button group boxes. Select the COM port from the ***Connection*** drop-down list box.

Typical configuration is shown below:

Sartorius LC3201D Port setup	
Baud Rate	19,200
Parity	Odd
Data Bits	7
Stop Bits	1
Flow Control	None

Exit by clicking the ***OK*** button to save your work. If you click the ***Cancel*** button, your settings will be forgotten.

Settings

The first section of the Settings property page is the ***Level Limits*** group box. This group box contains a ***Full Level*** and ***Empty Level*** limit (in grams). This driver has a digital input channel that will be set to TRUE whenever the current scale reading is above the ***Full Level*** setting; similarly, this driver also has another digital input channel that will be set to TRUE whenever the current scale reading goes below the ***Empty Level*** setting.

The ***Sample Frequency*** text box specifies the number of times per second the scale will be sampled for data.

Diagnose

For diagnostic purposes, a simple control screen has been provided. Once the system is set up, the ***Diagnose*** property pages can be used to check operation of the system. Clicking the ***Data*** button updates the screens.

In addition, a simple terminal emulator has been included to allow the user to send any desired data to the device and observe the response (if any). To do this, type the desired command in the ***Send:*** text box and click the ***Send*** button. Any response will be displayed next to the ***Rx:*** text.

Available Channels

Communication to the Sartorius LC3201D is accomplished through channels. Once the driver is installed and configured, the necessary channels are made available to the system. Please see Appendix A for detailed channel availability.

VIP Interface setup

This device does not need to be configured. This driver provides a fixed number of digital output, analog output, text output, and device control output channels that are used to exchange data between Cell Assistant TestPlans and external Virtual Instrument Panel (VIP) applications. VIP's are discussed in Chapter 10. This device (and hence all of these channels provided by this driver) are NOT connected to an actual device.

Available Channels

Communication with Virtual Instrument Panels is accomplished through channels. Please see Appendix A for detailed channel availability.

Chapter 5 - Channels

Click the ***Channels*** tab on the TestSystem property sheet. In order to proceed, you must enter (or have entered) the proper configuration level password. This will show the ***Channels*** property page.

Channel List

The most prominent feature of this screen is the list of currently defined channels. This box shows all of the channels that are currently defined, the hardware device and device channel each channel is "bound" to, and the channel units label. The channel list is a standard Windows list control. Initially, this list is sorted by channel name. The sorting method can be changed by pressing any of the buttons at the top of the list columns. For example, to sort the channel list according to hardware device of each channel, press the ***Device*** button at the top of this column. To restore the initial sort order (i.e. by channel name), press the ***Name*** button at the top of this column. At the present time, the channel list can be sorted by ***Name***, ***Device***, or ***Device channel***. It cannot be sorted by ***Units***.

The width of each column can be adjusted as needed in order to improve viewing of the channel data.

When sorting by ***Device*** or ***Device channel***, a ¶ symbol may appear before some items in the ***Device channel*** column. This indicates a duplicate channel; that is, one or more channels have been created and "bound" to the same device and device channel. Perhaps this was done intentionally, in which case, the operator can ignore the ¶ symbol; however, this is most likely due to an error that occurred when creating the channel. This type of mistake can easily occur when creating a TestSystem with hundreds of channels. Anyway, use this feature periodically to check for accidental creation of duplicate channels.

A count of the currently displayed channels is displayed just below the channel list. By applying various Display Filters (described below), you can easily count the total number of channels in the TestSystem or the total number of channels of any type.

Display Filter

Because the list of defined channels can get quite large, a filter feature is included which allows you to filter out the channels that you don't wish to see. This feature is controlled in the ***Display Filter*** group box.

Selecting the ***All*** option button shows all defined channels. Selecting the ***Selected*** option button allows you to determine which channels you wish to see.

Once the ***Selected*** option button has been selected, selecting the ***Input*** option button will allow you to see only the input channels. The types of input channels that are shown is further determined by which of the channel type check boxes are selected. One, some, or all boxes can be checked. Selecting the ***Output*** option button will allow you to see only the output channels.

Adding Channels

To add a channel, click the ***Add*** button (you must be logged in at the configuration level). This brings up the ***Add Channel*** dialog box. Select the type of channel you wish to add by selecting the down arrow button to the right of the ***Channel Type*** list box (channel types are described below). Then type

in the desired channel name in the ***Channel Name*** input box. The channel name can contain no spaces and can contain up to 255 characters. You may want to put a short prefix on your channel names to help identify what type they are. This often helps when writing a TestPlan. Typical prefixes are as follows:

"aic" for analog input channel (i.e. aicTorque)
"aoc" for analog output channel (i.e. aocThrottleSetpoint)
"avg" for average channel (i.e. avgFuelFlow)
"clc" for calculated channel (i.e. clcBSFC)
"ctr" for counter channel (i.e. ctrTestCycles)
"dic" for digital input channel (i.e. dicDynoCoolantPressure)
"doc" for digital output channel (i.e. docEmergStop)
"dco" for device command output channel (i.e. dcoDynoMode)
"dsi" for device status input channel (i.e. dsiNOxAnalRng)
"dde" for DDE channel (i.e. ddeExcel A1)
"tic" for text input channel (i.e. ticOperatorComment)
"tmr" for timer channel (i.e. tmrEngineRunTime)
"toc" for text output channel (i.e. tocEmissionsBench)

Click ***OK*** to dismiss the ***Add Channel*** dialog. The appropriate channel configuration dialog box will now be displayed

Removing Channels

To remove a channel, select a channel in the list (by clicking on an item in the ***Name*** column) and then click the ***Remove*** button (you must be logged in at the configuration level). An "Are you sure?" dialog will be displayed giving you a chance to change your mind. Click the ***OK*** button to delete this channel; otherwise, click the ***Cancel*** button.

Renaming Channels

To rename a channel, select a channel in the list (by clicking on an item in the ***Name*** column) and then click the ***Rename*** button (you must be logged in at the configuration level). This brings up the ***Rename Channel*** dialog box. Enter the ***New Channel Name*** and click the OK button to rename the channel; otherwise, click the Cancel button.

Clipping Channel Names

When typing in channel names in a TestPlan, it is easy to make mistakes such as misspelling the channel name and/or forgetting the channel name (because it is lengthy). The Windows system clipboard can easily be used to copy a channel name from the channel list to the system clipboard. Any of the standard "paste" mechanisms can then be used to insert the channel name into the TestPlan. To save a channel name to the system clipboard, highlight the desired channel name by clicking on the name of the channel in the Name column of the channel list. Then press the >> ***Clipboard*** button. The channel name can now be pasted into the TestPlan (or into any other application) by selecting ***Paste*** from the ***Edit*** menu or by using the short-cut key Ctrl+V.

Configuring Channels

This section will describe channel configuration items that are common to most channels. Following is a brief description of the channel types with channel configuration items that are unique to those channels.

The channel configuration dialog is displayed immediately after a new channel is added and the ***Add Channel*** dialog is dismissed by clicking the ***OK*** button. The configuration of an existing channel can also be changed by selecting a channel in the channel list (by clicking on an item in the ***Name*** column) and then clicking the ***Modify*** button (you must be logged in at the configuration level).

The name of the channel is displayed in the title bar of each channel configuration dialog box. The first item to configure is the source of the data going to or coming from the channel. Click the down arrow to the right of the ***Device*** drop-down list box in the ***Source*** group box. This will display a list of the hardware devices that the system is configured for. If you do not see the device you want, you will need to install it. See Chapter 4 - Hardware Devices: Adding Hardware Devices.

Once you have selected the device that will be the source of information for the channel you are configuring, you will need to select the channel, from that device, that presents the desired information. To do this, click the down arrow to the right of the ***Channel*** drop-down list box. This will display a list of the channels that the hardware device driver is configured for. These channels are hard coded into the device driver and cannot be changed or added to (except in the case of the Daytronic System 10). If the channel you want is not on the list, first be sure that you are looking for the correct channel type. For instance, you may be configuring an analog input channel when the channel you want is really a digital input channel. A list of the channels for each driver is given in Appendix A - Device Channels. If you still can't find the channel, call Dyne Systems for assistance.

At the present time, the sample rate for most channels is determined by the device driver and is selected there. The ***Sample Rate*** text box displays the sample rate determined in the driver and cannot be edited. The other alternative is "Polled" sampling. This means that the channel is not automatically updated. It will only be read and updated if it is used within a TestPlan. It is not recommended to configure a limit on a polled channel. Polled channels are used to get data that is only needed occasionally. By not reading the channel periodically, valuable processor time can be saved.

Some channels already come into Cell Assistant in engineering units. Others come in as raw signals (such as voltage, frequency, current, etc.). These must usually be converted into engineering units. This is done with the items in the ***Data Conversion*** group box. The first item is the ***Engineering Units*** input box. Type in the desired units (such as deg. F, deg. C, RPM, lb-ft, pounds per hr, etc..., maximum of 63 characters) or select from the drop-down list. Please note that the engineering units for the drop-down list are stored in a text file named EngUnits.txt which is located in the CA4W subdirectory. Other units can be added by opening this file with an ASCII text editor (such as Microsoft Notepad, which is included on most computers), adding the desired unit, and resaving the file. The changes will then appear after Cell Assistant has been started or restarted.

Next choose the number of decimal places that you want the data to be rounded to in the ***Decimal Places*** drop-down list box. This determines the rounding used when the channel updated. The data is internally stored with this precision; thus, this is the precision that will be displayed whenever this channel is displayed in a dialog box and/or recorded to a data file or Flight Recorder file.

If the channel needs to be converted mathematically, the slope and offset of the conversion formula (Y = m X + b where m is Slope and b is Offset) are entered in the ***Slope m*** and ***Offset b*** input boxes. Linearity is assumed. If you wish to quickly look at the channel without applying the slope and offset, check the ***Show Raw Data*** check box below the ***Data Conversion*** group box. This will internally set the slope to one and the offset to zero and gray out the Slope and Offset list boxes. To return to the entered slope and offset, uncheck the ***Show Raw Data*** check box.

Channel Types

Physical Vs. Computed Channels

Cell Assistant has two kinds of channels: physical channels and computed channels. Physical channels are channels that get data directly from the outside world (e.g. from an external hardware device). Examples are analog input channels, digital input channels, device status input channels, and text input channels. Computed channels are derived from physical channels. Examples are average channels, calculated channels, and counter channels.

Internally, Cell Assistant is constantly reading data from physical devices and updating all necessary physical channel values. It is also constantly recomputing all computed channels. During each update cycle, physical channels are ALWAYS updated before computed channels are updated since physical channels are usually the input channels to most computed channels.

Analog Input Channel

This is an input (to Cell Assistant) channel that can have more than two values (as opposed to a digital input, which can have only two values: true and false).

Use in a TestPlan

To set a variable equal to the value of a channel, use the following code:

```
X = aicChannelName.Value
```

or

```
X = aicChannelName
```

Note that aicChannelName and aicChannelName.Value are double precision floating-point numbers.

To set a string variable equal to a channels units (as set in the ***Channel Modify*** dialog box), use the following code:

```
Text$ = aicChannelName.Units
```

Note that aicChannelName.Units is a string.

Some transducers exhibit zero drift. In these cases, it may be desirable to zero the channel before a test starts. To zero an analog input channel, use the following code

```
aicChannelName.Autozero
```

This will cause the channel to be offset enough for the channel to read zero at the instant the line was executed. When the TestPlan is over, the offset is reset to the value determined by calibration and the ***Data Conversion*** group box in the ***Channel Modify*** dialog box. The author of the TestPlan must be

sure that the transducer is actually reading zero when the line is executed or future readings will be inaccurate.

Analog Output Channel

This is an output (from Cell Assistant) channel that can have more than two values (as opposed to a digital output, which can have only two values: true and false).

Use in a TestPlan

To set the value of an analog output channel equal to a variable, use the following code:

```
aocChannelName.Value = X
```

or

```
aocChannelName = X
```

Note that X should be a double precision number.

To set a string variable equal to a channels units (as set in the ***Channel Modify*** dialog box), use the following code:

```
Text$ = aocChannelName.Units
```

Note that aocChannelName.Units is a string.

Average Channel

This is a channel that is the average of another channel. It is typically used to smooth data. At other times, it is used literally to compute the mathematical average of a series of data points such that the TestPlan writer doesn't have to (since channel data averaging could also be implemented in DSBasic). The source for an average channel is always another channel that has already been configured. That channel is selected from the ***Channel*** drop-down list box in the ***Source*** group box.

In addition to the source channel, the method of averaging must also be selected from the ***Method*** drop-down list box in the ***Averaging*** group box. Presently, the only method of averaging is "Shift Register". This means that a register (i.e. a block of memory) is established. The number of values in the register is determined by the internal sample rate of the software and the specified averaging ***Time*** interval for this channel. The shift register is initially filled with the value of zero. The register is continuously loaded with values of the source channel. Each time a new value of the source channel is read, the new value is added to the register, the oldest value in the register is discarded and the values in the register are summed and divided. This results in a true average reading.

Use in a TestPlan

To set a variable equal to the value of a channel, use the following code:

```
X = avgChannelName.Value
```

or

```
X = avgChannelName
```

Note that avgChannelName and avgChannelName.Value are double precision floating-point numbers.

To set a string variable equal to a channels units (as set in the ***Channel Modify*** dialog box), use the following code:

```
Text$ = avgChannelName.Units
```

Note that avgChannelName.Units is a string.

To reset the buffer of an average channel, use the following code:

```
avgChannelName.Reset
```

This causes the entire contents of the shift register to be filled with zeros again (just like at the start of data acquisition). After one ***Time*** interval has elapsed, the average will be fully "charged up".

Calculated Channels

Calculated channels are channels that are the result of a calculation. Typically this calculation involves other channels. For instance, if fuel flow rate were being read in kilograms per hour, it may be desirable to add a channel that calculates the flow rate in pounds per hour.

The ***Function Definition*** group box in the channel configuration dialog box has three option buttons - ***Default TestPlan, Global TestPlan,*** and ***System Function***. Click the appropriate button based on the type of calculated channel you want to create. Each type is described in the sections that follow.

System Functions

A System Function is a calculation that is pre-programmed in Cell Assistant. These are calculations that are commonly used in the industry. Since these require less processor time to calculate, it is a good idea to use them whenever possible. At the present time, the following system functions are implemented.

J607 Correction Factor (Dew Point)

This function calculates the SAE J607 correction factor for torque and power correction using dew point as the measure of humidity.

The three channels needed for this function are:

Barometric Pressure in kPa
Dew Point in degrees C
Intake Air Temperature in degrees C

The actual formula used is:

$$CorrectionFactor = \frac{101}{B - E} \times \sqrt{\frac{273 + CAT}{288.6}}$$

where: B = Barometric Pressure in kPa
CAT = Intake Air Temperature in degrees C
E = Vapor Pressure in kPa

$$E = Ae^{B\left(\frac{t_{dp}}{t_{dp}+238.3}\right)}$$

where: t_{dp} = Dew Point in degrees C
A = 0.61078 kPa
B = 17.2694

J1349 Correction Factor (Dew Point)

This function calculates the SAE J1349 correction factor for torque and power correction using dew point as the measure of humidity.

The three channels needed for this function are:
Barometric Pressure in kPa
Dew Point in degrees C
Intake Air Temperature in degrees C

The actual formula used is:

$$CorrectionFactor = 1.18 \times \left(\frac{99}{B_o - E}\right) \times \sqrt{\frac{t + 273}{298}} - 0.18$$

where: B_o = Barometric Pressure in kPa
t = Intake Air Temperature in degrees C
E = Vapor Pressure in kPa

$$E = Ae^{B\left(\frac{t_{dp}}{t_{dp}+238.3}\right)}$$

where: t_{dp} = Dew Point in degrees C
A = 0.61078 kPa
B = 17.2694

J1349 Correction Factor (Relative Humidity)

This function calculates the SAE J1349 correction factor for torque and power correction using relative humidity as the measure of humidity.

The four channels needed for this function are:
Barometric Pressure in kPa
Relative Humidity in percent
Intake Air Temperature in degrees C
Ambient Air Temperature in degrees C

The actual formula used is:

$$CorrectionFactor = 1.18 \times \left(\frac{99}{B_o - E}\right) \times \sqrt{\frac{t + 273}{298}} - 0.18$$

where: B_o = Barometric Pressure in kPa
t = Intake Air Temperature in degrees C
E = Vapor Pressure in kPa

$$E = \frac{\%\ RH}{100} \times Ae^{B\left(\frac{t_a}{t_a + 238.3}\right)}$$

where: t_a = Ambient Air in degrees C
A = 0.61078 kPa
B = 17.2694

ISO 3046-1 Correction Factor

This function calculates the ISO 3046-1 correction factor for torque and power correction.

The four channels needed for this function are:
Barometric Pressure in kPa
Relative Humidity in percent
Intake Air Temperature in degrees C
Temperature of the Relative Humidity sample in degrees C

The actual formula used is:

$$CorrectionFactor = \left(\frac{99}{P_b}\right)^{1.2} \times \left(\frac{AI + 273}{298}\right)^{0.6}$$

where: AI = Intake Air Temperature in degrees C

$P_b = B - \%RH * [T_1 + (T_2 * T_h) + (T_3 * T_h^2) + (T_4 * T_h^3) + (T_5 * T_h^4)]$

where: B = Barometric Pressure in kPa
%RH = Relative Humidity in percent
T_h = Temperature of the Relative Humidity sample in degrees C
T_1 = 0.620854
T_2 = 0.0390123
T_3 = 0.00188944
T_4 = 0.00000663369
T_5 = 0.000000657956

Degree F to Degree C Conversion

This function converts a temperature in degrees Fahrenheit to a temperature in degrees Celsius.

The one channel needed for this conversion is:
Temperature in degrees F

The actual formula used is: $T_c = (T_f - 32) * 5/9$

where: T_f = Temperature in degrees F
T_c = Temperature in degrees C

Degree C to Degree F Conversion

This function converts a temperature in degrees Celsius to a temperature in degrees Fahrenheit.

The one channel needed for this conversion is:
Temperature in degrees C.

The actual formula used is: $T_f = (T_c * 9/5) + 32$

where: T_c = Temperature in degrees C
T_f = Temperature in degrees F

Pounds to Kilograms Conversion

This function converts a mass in pounds (lb) to a mass in kilograms (kg). We realize that pounds are actually units of force but since this conversion is commonly used, it was included.

The one channel needed for this conversion is:
Mass in lb

The actual formula used is: $M_k = M_p * 0.4536$

where: M_p = Mass in lb
M_k = Mass in kg

Kilograms to Pounds Conversion

This function converts a mass in kilograms (kg) to a mass in pounds (lb). We realize that pounds are actually units of force but since this conversion is commonly used, it was included.

The one channel needed for this conversion is:
Mass in kg

The actual formula used is: $M_p = M_k * 2.205$

where: M_p = Mass in pounds
M_k = Mass in kilograms

Brake Specific Fuel Consumption

This function calculates Brake Specific Fuel Consumption (BSFC) from fuel consumption and power. The units depend on the units used for power and fuel consumption.

The two channels needed for this calculation are:
Fuel consumption
Power

The actual formula used is: BSFC = F/P

where: F = Fuel consumption
P = Power

PSI To Inches of Mercury

This function converts a pressure in pounds per square inch (PSI) to a pressure in inches of mercury (in. Hg) at 0° C.

The one channel needed for this conversion is:
Pressure in PSI

The actual formula used is: $P_i = P_p * 2.036$

where: P_p = Pressure in PSI
P_i = Pressure in in. Hg

Inches of Mercury to PSI

This function converts a pressure in inches of mercury (in. Hg) at 0° C to a pressure in pounds per square inch (PSI).

The one channel needed for this conversion is:
Pressure in in. Hg

The actual formula used is: $P_p = P_i * 0.4912$

where: P_p = Pressure in PSI
P_i = Pressure in in. Hg

Inches of Mercury to kiloPascals

This function converts a pressure in inches of mercury (in. Hg) at 0° C to a pressure in kiloPascals (kPa).

The one channel needed for this conversion is:
Pressure in in. Hg

The actual formula used is: $P_k = P_i * 3.386$

where: P_k = Pressure in kPa
P_i = Pressure in in. Hg

Pound Feet to Newton Meters

This function converts a torque in pound feet (lb ft) to a torque in Newton meters (Nm).

The one channel needed for this conversion is:
Torque in lb ft

The actual formula used is: $T_n = T_l * 1.3558$

where: T_l = Torque in lb ft
T_n = Torque in Nm

Newton Meters to Pound Feet

This function converts a torque in Newton meters (Nm) to a torque in pound feet (lb ft).

The one channel needed for this conversion is:
Torque in Nm

The actual formula used is: $T_l = T_n * 0.7376$

where: T_l = Torque in lb ft

T_n = Torque in Nm

Horsepower

This function calculates horsepower (HP) from torque and speed.

The two channels needed for this calculation are:
Torque in lb ft
Speed in RPM

The actual formula used is: $HP = N \times T/5252$

where: N = Speed in RPM
T = Torque in lb ft

Horsepower to kilowatts

This function converts power in horsepower (HP) to power in kilowatts (kW).

The one channel needed for this conversion is:
Power in HP

The actual formula used is: $P_{kW} = P_{HP} * 0.746$

where: P_{kW} = Power in kW
P_{HP} = Power in HP

kilowatts to Horsepower

This function converts power in kilowatts (kW) to power in horsepower (HP).

The one channel needed for this conversion is:
Power in kW

The actual formula used is: $P_{HP} = P_{kW} / 0.746$

where: P_{kW} = Power in kW
P_{HP} = Power in HP

Multiplication (2 channels)

This function multiplies any two channels together creating a product that is a new channel. This is a very general System Calculated channel and is used most often to multiply torque or power by a correction factor.

TestPlan Functions

TestPlan functions are functions that are defined in the TestPlan (i.e. written and implemented in the DSBasic language) rather than being pre-programmed into the system. You must first decide which TestPlan should contain the DSBasic code for the calculated functions. Your choices are the ***Default TestPlan*** or the ***Global TestPlan***. See Chapter 7 for more information on Default TestPlans and Global TestPlans. Typically, the implementation of a calculated function will go into the Global TestPlan so that it is available to all other TestPlans.

When you select the ***Default TestPlan*** or ***Global TestPlan*** option button, the appropriate drop-down list box will be enabled. You now have two choices - to use a function that is already defined in the respective TestPlan, or to create a new one. If the function that you wish to use is already defined in the respective TestPlan (or you think it is), click the down arrow on the right of the drop-down list box. This will display a list of the calculated channel functions presently in the respective TestPlan. Select the one you want from the list.

If you want to create a new one, type the name of the function in the drop-down list box. Then click the ***Add*** button just to the right of the list box. When you click this button, a couple of things happen. First of all, the text that you typed in the drop-down list box has been changed by adding "CLC_" to the beginning of it (unless you already typed this in). This is to assure that proper syntax is being followed. Secondly, a few lines of DSBasic code were added to the appropriate TestPlan (Default or Global). These lines create an empty shell into which you have to type the desired DSBasic code. These will be explained in the following example.

For this example, let's assume that we want to calculate horsepower from speed and torque. We have previously configured a channel for speed which we have named "aicDynoSpeed" ("aic" is to remind us that this is an analog input channel) and units are RPM, and a channel for torque, which we have named "aicDynoTorque" and units are pound feet. Click ***Add*** in the ***Channels*** property page and select "Calculated" in the ***Channel Type*** drop-down list box of the ***Add Channel*** dialog box. Type "clcPower" in the ***Channel Name*** input box. Click ***OK***.

Select the ***Global TestPlan*** option button, and type "Power" in the drop-down list box. Then click the ***Add*** button. The word "Power" in the drop-down list box will change to "CLC_Power". In the Engineering Units input box type "Horsepower" and click ***OK***. You should now be looking at the Global TestPlan. It is a good idea to name the procedure the same as the channel (or at least close) so you can easily associate the two later.

At the top of the Global TestPlan, under the `Rem {{DECLARATIONS}}` line, the following line will have been added:

```
Declare Function CLC_Power () As Double
```

The function declaration has to be there per DSBasic syntax.

At the bottom of the TestPlan the following will have been added:

```
Function CLC_Power () As Double
      ' Insert calculation here
      CLC_Power = 0
End Function
```

This code must now be modified to insert the proper calculation. When this is done, the code will look like the following:

```
Function CLC_Power () As Double
      ' Insert calculation here
      CLC_Power = aicSpeed * aicTorque / 5252
End Function
```

If desired, these functions can be written manually when composing a TestPlan. Just remember to do the function declaration at the beginning of the TestPlan and follow the naming format shown above. See the DSBasic User's Guide for more information.

TestPlan calculated functions should be short and simple

TestPlan calculated functions are implemented in DSBasic just like the main body of the TestPlan code; hence, all DSBasic functions are available for use in a TestPlan calculated function. In practice, however, many restrictions exist. TestPlan calculated functions are NOT called by TestPlan code; they are called a regular intervals (20 Hz actually) by the Cell Assistant data acquisition subsystem. In order to NOT destroy the real-time data acquisition timing of the system, all TestPlan calculated functions should be as short as possible. They typically should only be a few lines of code that perform some arithmetic on several other channels to produce the calculated channel output. Perhaps a few conditional statements (e.g. If-Then-Else) will also be used to handle special cases. But that should be the extent of a TestPlan calculated function.

Message boxes, popup dialog boxes, lengthy loops, OLE automation, file I/O, etc … should NEVER BE USED in a TestPlan calculated function since they can potentially take too long to execute and, hence, disturb the timing of the Cell Assistant data acquisition subsystem.

Division by Zero

When creating a calculated channel using a TestPlan function, it is a good idea to include code to avoid divide by zero errors. A good example is if you are manually calculating brake specific fuel consumption (which, of course, you don't have to do because this is a system function). The code may look something like this:

```
Function CLC_BSFC () As Double
      ' Insert calculation here
      CLC_BSFC = aicFuelFlow / aicHorsepower
End Function
```

Unfortunately, there are times when aicHorsepower will be zero. Actually, this will happen anytime the engine is off. The code should be written to deal with this. The following illustrates a good method:

```
Function CLC_BSFC () As Double
      ' Insert calculation here
      If aicHorsepower = 0 Then
            CLC_BSFC = 0
      Else
            CLC_BSFC = aicFuelFlow / aicHorsepower
      End If
End Function
```

Counter Channel

Counter channels are meant to count slowly changing digital input channels. Slowly changing means that when the digital input channel changes state, it should not change again for at least one sample period. If it does change, the event may not be counted. For example, if a digital input channel is getting data from a PLC at a rate of two Hertz, the signal must remain unchanged for at least ½ second. If it's being sampled at one Hertz, the signal must remain unchanged for at least one second.

The ***Counter*** group box contains three configuration items. Select the ***Up*** option button if you want the counter to count up. Select the ***Down*** option button if you want the counter to count down. The counter can be preset to a value when the preset command is issued in a TestPlan. Enter the value that you want the counter to be preset to in the ***Preset*** input box. Please note that the ***Preset*** value must be

an integer (no decimal places) and that the ***Preset*** value is the number of counts, not the channel value after being modified by m and b.

Each time Cell Assistant detects a leading edge (transition from a zero to a one), it will increment the counter by one. If you want the counter to increment by some other number, enter it in the ***Slope m*** input box. The ***Offset b*** input box also functions as you would expect.

Note that configuring a counter channel to count a polled digital input channel should be avoided because the digital input channel will not change unless it is polled from within a TestPlan. Before it can add (or subtract) another count, it must be polled during a low state, and then again during a high state because the counter is incremented when state changes from low to high.

Use In TestPlans

The following example will use "tmrCounterChannel" as the name of the counter channel.

To cause the counter to start counting events, use the following code:

```
tmrCounterChannel.Start
```

To cause the counter to stop counting events, use the following code:

```
tmrCounterChannel.Stop
```

The channel will stop counting but will retain the value it had when stopped.

To reset the counter to a value of zero, use the following code:

```
tmrCounterChannel.Reset
```

To preset the counter to the value entered in the ***Channel Modify*** dialog, use the following code:

```
tmrCounterChannel.Preset
```

To preset the counter to any arbitrary value (e.g. 400 in this example), use the following code:

```
tmrCounterChannel.Preset 400
```

Device Status Input Channel

This channel is specially designed to get status information from an external device.

Device Control Output Channel

This channel is specially designed to send commands to an external device.

Digital Input Channel

This is an input (to Cell Assistant) channel that can have two values, one (true) or zero (false). This type of channel has three unique configuration items. Two are configured in the ***Display Information*** group box. They are ***ON Text*** and ***OFF Text***. In these input boxes, enter a text string (maximum of 63 characters each) that describes the on state and off state of the channel. For example, if a dyno coolant pressure switch were used, the ON Text might be "Pressure Normal" and the OFF Text may be

"Pressure Low". As a convenience, the ***Invert Logic*** check box is included. Checking this box will invert the logic in case you weren't sure of the logic when the channel was configured.

Use In TestPlans

The following example will use "dicChannelName" as the name of the digital input channel.
To set a variable equal to the value of a channel, use the following code:

```
X = dicChannelName.Value
```

or

```
X = dicChannelName
```

Note that X should be a Boolean value.

To set a string variable equal to the ON or OFF Text of a digital input channel, use the following code:

```
X$ = dicChannelName.Text
```

The current state of the digital input channel determines whether the ON or OFF Text is returned. If the current state of the channel is true, the ON Text is returned; if the current state of the channel is FALSE, the OFF Text is returned.

Digital Output Channel

This is an output (from Cell Assistant) channel that can have two values, one (true) or zero (false). This type of channel has three unique configuration items. Two are configured in the ***Display Information*** group box. They are ***ON Text*** and ***OFF Text***. In these input boxes, enter a text string that describes the on state and off state of the channel. For example, if a fuel solenoid were used, the ON Text might be "On" and the OFF Text may be "Off". As a convenience, the ***Invert Logic*** check box is included. Checking this box will invert the logic in case you weren't sure of the logic when the channel was configured.

Use In TestPlans

The following example will use "docChannelName" as the name of the digital output channel.

To set the value of a digital output channel equal to a variable, use the following code:

```
docChannelName.Value = X
```

or

```
docChannelName = X
```

Note that X should be a Boolean value.

To set a string variable equal to the ON or OFF Text of a digital output channel, use the following code:

```
X$ = docChannelName.Text
```

The current state of the digital output channel determines whether the ON or OFF Text is returned. If the current state of the channel is true, the ON Text is returned; if the current state of the channel is FALSE, the OFF Text is returned.

DDE Input Channel

This channel is designed to get data from another Windows application that supports DDE (Dynamic Data Exchange). Input boxes are provided for the three pieces of information that are needed to establish a DDE link. They are ***Application Name***, ***Topic Name***, and ***Item Name***. These strings can be typed in, or, if link information has been copied to the Windows clipboard from another application, clicking the ***Paste Link*** button will automatically insert the information. Our experience has shown that multiple DDE channels can consume a lot of processor time.

As an example, the following DDE channel configuration settings can be used to create a channel that inputs data from the upper-left most cell (i.e. Row 1, Column 1) in a spreadsheet.

Application Name: Excel
Topic Name: DdeInputDataSource.xls
Item Name: R1C1

Clearly, the ***Application Name*** is the executable name of the spreadsheet program that you are using - in this case, Excel.exe. The ***Topic Name*** is the name of the spreadsheet file. The ***Item Name*** it the Row-Column (RxCx) address of the data source.

Text Input Channel

Text input channels allow text strings to be used in Cell Assistant just like other input channels. These are used to get things like status information from devices and operator comments.

Text Output Channel

Text output channels provide the ability to send text messages to external devices and VIP screens.

Timer Channel

This type of channel is used to keep track of time. Only three items can be configured for a timer channel. Select the ***Up*** option button if you want the timer to count up. Select the ***Down*** option button if you want the timer to count down. The timer can be preset to a value when the preset command is issued in a TestPlan. Enter the value that you want the counter to be preset to in the ***Preset*** input box. Please note that the preset value must be an integer (no decimal places).

The resolution of a timer channel is one second and the value is always in seconds (rather than hours:minutes:seconds).

Use In TestPlans

The following example will use “tmrTimerChannel” as the name of the timer channel.

To cause the timer to start counting seconds, use the following code:

```
tmrTimerChannel.Start
```

To cause the timer to stop counting seconds, use the following code:

```
tmrTimerChannel.Stop
```

The channel will stop counting but will retain the value it had when stopped.

To reset the timer to a value of zero, use the following code:

```
tmrTimerChannel.Reset
```

To preset the timer to the value entered in the ***Channel Modify*** dialog, use the following code:

```
tmrTimerChannel.Preset
```

To preset the timer to a value other than the value entered in the ***Channel Modify*** dialog, use the following code:

```
tmrTimerChannel.Preset 60
```

The value of the channel will be set to 60.

Chapter 6 - Limits

Please note that to work with Limits, you have to be logged on in the configuration level.

Creating a New Limit

Click the ***Limits*** tab on the TestSystem property sheet to expose the ***Limits*** property page. The ***Defined Limits*** list box shows all of the limits that have already been defined. To add a new one, click the ***Add*** button. Type the name of the limit into the edit box (spaces are not allowed) and click ***OK***. If you wish to remove a limit that's on the list, choose it and then click the ***Remove*** button.

Examples of limit names may be "LowOilPressureWarning", "LowOilPressureShutdown", "SoftShutdown", or "HardShutdown".

Monitored Channel

Enter the name of the channel that you wish to have the limit on. Typically the easiest way to do this is to click the down arrow to the right of the ***Monitored Channel*** box and choose the desired channel from the list. Please note that a single channel can have more than one limit on it.

Limit Type

It is now necessary to choose the type of limit desired. In the ***Limit Type*** group box is a set of option buttons for selecting the desired limit type. These are described below.

Trigger Low option button

Limit will trigger if the value of the channel becomes less than the setpoint value entered into the ***Low Limit*** edit box. Be sure to fill in the desired limit value into the ***Low Limit*** edit box.

Trigger High option button

Limit will trigger if the value of the channel becomes greater than the setpoint value entered into the ***High Limit*** edit box. Be sure to fill in the desired limit value into the ***High Limit*** edit box.

Trigger In-Range option button

Limit will trigger if the value of the channel becomes less than or equal to the High Limit setpoint entered into the ***High Limit*** edit box AND greater than or equal to the Low Limit setpoint entered into the ***Low Limit*** edit box. Be sure to enter the desired limit values into the ***High Limit*** and ***Low Limit*** edit boxes.

Trigger Out-of-Range option button

Limit will trigger if the value of the channel becomes less than the Low Limit setpoint entered into the ***Low Limit*** edit box or if the value of the channel becomes greater than the High Limit setpoint entered into the ***High Limit*** edit box. Be sure to enter the desired limit values into the ***High Limit*** and ***Low Limit*** edit boxes.

Trigger Type

The trigger type of each limit determines when and how often the DSBasic limit handler is executed when a limit trips. The choices are Level and Edge triggered. They are described below.

Level Triggered option button

The DSBasic limit handler will be repeatedly called as long as trip conditions persist for a given limit as follows. When the limit task determines that trip conditions exists for a limit, the DSBasic limit handler is called. If this handler fails to correct the cause of the trip and doesn't disable the limit, the limit task will evaluate all of the other limits and, in turn, will get back to this limit and see that trip conditions still exist. Thus, the limit handler will be called again. This series of events will continue indefinitely until the trip condition is cleared or the limit handler disables this limit.

Edge Triggered option button

For an edge triggered limit, the DSBasic limit handler will only be called at the onset and offset of trip conditions. The limit handler will be called once when the trip condition is first detected. If will NOT be called again until the trip condition clears itself. Thus, the DSBasic limit handler will be called exactly two times when an edge triggered limit trips.

The DSBasic limit handler can easily distinguish if it is being called at the onset or the offset of a trip condition and, thus, execute a unique set of instructions for each condition. This is explained in a later section - **Using Limits in TestPlans.**

Priority

A limit is either a high priority limit or not (if it's not, it will be referred to as a low priority limit).

Channels with high priority limits and low priority limits are always being checked. If a low priority limit is triggered, the limit procedure (described below) is executed. Once this happens, the high priority limits continue to be checked, but if another low priority limit is triggered, the corresponding procedure will not be executed until the currently executing low priority limit procedure has finished executing. If, however, a high priority limit is triggered, execution of the low priority limit procedure will be suspended and the high priority limit procedure will be executed. If another high priority limit is triggered, it will be ignored until the currently executing high priority limit procedure has finished executing.

Procedure

Once a limit is triggered, Cell Assistant executes a procedure or function contained in either the Default or Global TestPlan (see Chapter 7 - TestPlans). If you wish to create a new procedure to be executed when the limit is triggered, first decide if you want the limit procedure to reside in the ***Default TestPlan*** or the ***Global TestPlan*** and click the appropriate button. This will enable the appropriate drop-down edit box and ***Add*** button. Type the desired name in the edit box, then click the ***Add*** button. This will cause an empty procedure to be written into the Default or Global TestPlan. You must now go into the TestPlan editor and write the code needed to perform the desired action.

Many times, a new limit procedure is not needed. To use a limit procedure that has already been defined, click the down arrow to the right of the appropriate edit box. This will cause a list of the existing procedures to be displayed. The ***Default TestPlan*** list will only contain limit handlers that

currently exist in the Default TestPlan; similarly, the ***Global TestPlan*** list will only contain limit handlers that currently exist in the Global TestPlan. Choose the desired procedure. The list will disappear leaving the name of the desired procedure in the box. For example, if eight exhaust gas thermocouples had high limits we would probably want the same action to occur no matter which channel triggered a limit. To do this, we create a limit procedure when configuring a limit on the first thermocouple channel. For each of the seven subsequent limits, we select the same procedure from the list of limit procedures. In some cases, every limit may execute the same procedure when triggered.

Using Limits in TestPlans

The following sections describe how to control Limits from within a DSBasic TestPlan. For all of the following sections, the word LimitName is used. This is a generic way to refer to the actual name of a limit. An actual Limit Name would be something like “LowOilPressure” or “HiHeadTemp”.

Enable Method

When a TestPlan starts, all limits are disabled. They must be individually enabled in the TestPlan. To enable a limit, use the method “LimitName.Enable”. For example:

```
LowOilPressure.Enable
```

Disable Method

To disable a limit use the method “LimitName.Disable”. For example:

```
LowOilPressure.Disable
```

Low Property

To determine (read) the low value of a limit (either a low limit, an in-range limit, or an out-of-range limit), use the command “Variable = LimitName.Low”. For example:

```
OldLimit = LowOilPressure.Low
```

To set the low value of a limit (either a low limit, an in-range limit, or an out-of-range limit), use the command “LimitName.Low = Variable”. For example:

```
LowOilPressure.Low = NewLimit
```

The low value can be modified as often as needed by a running TestPlan. It will always be set back to the default configured value when a TestPlan is started. These commands affect the currently selected limit set so you must keep track of which limit set is currently selected. Limit sets are explained in another section that follows.

High Property

To determine (read) the high value of a limit (either a high limit, an in-range limit, or an out-of-range limit), use the command “Variable = LimitName.High”. For example:

```
OldLimit = HighOilPressure.High
```

To set the high value of a limit (either a high limit, an in-range limit, or an out-of-range limit), use the command “LimitName.High = Variable”. For example:

```
HighOilPressure.High = NewLimit
```

The high value can be modified as often as needed by a running TestPlan. It will always be set back to the default configured value when a TestPlan is started. These commands affect the currently selected limit set so you must keep track of which limit set is currently selected. Limit sets are explained in another section that follows.

Function Property

To determine (read) the function that will be run when a limit is exceeded, use the command "Variable$ = LimitName.Function", where Variable$ is a string. For example:

```
WhichFunct$ = LowOilPressure.Function
```

Channel Property

To determine (read) the channel that the limit is checking, use the command "Variable$ = LimitName.Channel", where Variable$ is a string. For example:

```
WhichChan$ = LowOilPressure.Channel
```

Status Property

To determine if trip conditions exist for any limit, use the command "Variable = LimitName.Status", where Variable is a Boolean which will be TRUE if trip conditions exist, and FALSE if trip conditions do not exist. This will check the limit and return the correct value even if the limit is disabled. For example:

```
TripConditionStatus = LowOilPressure.Status
```

This property merely reports whether or not trip conditions exist for a limit. If trip conditions do exist, it will NOT cause an actual trip to occur (i.e. execute the appropriate limit handler code). This is useful in situations where it is known that a particular limit will remain in a tripped condition for a long period of time. The TestPlan may disable the limit such that it won't continuously trip; yet, the TestPlan can still leisurely monitor for the trip condition to go away. Afterwards, the limit can be re-enabled.

IsTripped Property

This method returns the current trip state of a limit; that is, the result of the last time the limit task evaluated this limit. For example:

```
TripState = LowOilPressure.IsTripped
```

This property is subtly different from the `Status` property. The state of the `IsTripped` property will persist until the limit task gets around to re-evaluating the respective limit even though the trip conditions for this limit may have already changed.

The DSBasic Limit handler

Each limit typically has a DSBasic limit handler that is called whenever a limit trips. This handler can be very simple (e.g. merely disable the limit and/or or stop the running TestPlan) or it can be complex (e.g. attempt to determine the cause of the trip condition and try to rectify the cause). When and how often the limit handler is called depends on the trigger type of each limit as explained in previous sections.

Limit Handler Format

When a limit procedure is added (see Limits Page section), the following lines of code are automatically added to the Global or Default TestPlan (as selected by the operator).

```
Function LIM_MyLimit () As Integer
      If ThisLimit.IsTripped Then
            ' TODO: Add code to handle the onset of an edge-triggered
            '       limit or a level-triggered limit here.
      Else
            ' TODO: Add code to handle the offset of an
            '       edge-triggered limit here.
      End If
      LIM_MyLimit = 0
End Function
```

Each limit handler function must return a value. This is accomplished by the `LIM_MyLimit = 0` statement at the end of the function. Valid return codes and their meaning are explained in the next section.

The first thing this code stub does is to determine if this limit handler has been called at the onset of an edge-triggered limit (or any occurrence of a level-triggered limit) or has been called at the offset of an edge-triggered limit. There are only two possibilities. Code comments are automatically added that explain where to put your code to handle each condition.

If a limit is (and probably always will be) a level-triggered limit, this code can be deleted since the "offset" case does not exist for a level-triggered limit.

Limit Handler Return Value

A limit handler must always set a return value before exiting. This return value determines what the limit task will do when this handler returns. Valid return codes are listed below.

Return code = 0	TestPlan continues to run.
Return code = 255	Stop the currently running TestPlan.
All other codes	[reserved for future use]

A limit handler should never attempt to end a running TestPlan (i.e. execute the "`End`" statement) by itself; instead, it should return a value of 255 and let the limit task stop the TestPlan. If a TestPlan is stopped (again, by executing the "`End`" statement) within the context of a limit handler (or TestPlan calculated function also), the results are unpredictable.

ThisLimit object

Each DSBasic limit handler function is typically associated with a specific limit. Knowing this, a limit handler can access methods and properties for the associated limit object (e.g. check the trip state of this limit, modify a setpoint value, disable the limit, etc …). Once a limit handler contains references to a specific limit object, it can no longer be used as a limit handler for any other limit. The ability to use a DSBasic limit handler for more than one limit is provided by the "`ThisLimit`" object as explained below.

The "`ThisLimit`" object is identical to a limit object. Most (but not all) of the methods that are available for a limit object are also available for the `ThisLimit` object. This object only exists

during the context of a limit handler. The limit task creates this object and sets it to reference the current limit prior to running the limit handler. This object allows methods of the currently active limit to be referenced in the limit handler without specifically referencing any particular limit. The following methods are available.

```
ThisLimit.Enable
ThisLimit.Disable
ThisLimit.High
ThisLimit.Low
ThisLimit.IsTripped
ThisLimit.Status
```

Limit Sets

As explained in previous sections, limits are created and configured in the Limits property page. The high and low setpoint values can be modified as needed by any running TestPlan. In some applications, where the TestPlan routinely makes the same modifications to certain limits, a better implementation is to use Limit Sets. Every TestSystem starts with a "Default" set of limits. This is the limit set that is created and modified in the Limits page. The Limit Set feature of Cell Assistant allows you to make copies of the "Default" limit set (or any other previously created limit set). You can then make minor or major changes to any or all of the setpoint values of the new limit set. Each limit set must be given a unique name. The "Default" set is always named "Default" and cannot be changed. The "Default" limit set is automatically selected when any TestPlan starts. Afterwards, the TestPlan can change limit sets as often as needed.

If you have no application that requires the use of multiple limit sets, you will be unaffected by the existence of this feature. If you want to create and/or modify limit sets, click the ***Limit Sets …*** button on the Limits page. The Limits Sets page will be displayed.

Limit Sets Page

This page is used to add, remove, and modify limit set data. High and low setpoint data for each limit and limit set is displayed in a rectangular grid. All currently defined limits are listed in the first column. Each limit spans one or two rows depending on the limit type. The list of currently defined limit sets are listed across the top row. Initially only the "Default" limit set column is shown. It is always shown in yellow and it will not scroll off of the screen. It is frozen in place such that other limit sets can be scrolled into position next to it for easy comparison of setpoint values.

To create a new limit set, first select the limit set that is most similar to the new one that you want to create by clicking on the appropriate limit set name. Initially only the "Default" limit set exists so click on the cell that says "***Default***". This will select the entire column. Now click the ***Duplicate*** button. Enter a unique name for this new limit set in the dialog box that appears and click the ***OK*** button. The new limit set column will be inserted to the right of the selected limit set. All setpoint values will be duplicated also. You can now edit various cells in this new column to finalize the new limit set.

Notice that as you read across any row, you are looking at the setpoint values for a specific setpoint (high or low) for a specific limit. All values in any row that match the same value in the "Default" limit set are shown in a lighter gray text color. This makes it easy to see at a glance how a given limit

set deviates from the “Default” set. If you change a setpoint value, it will then be shown in darker black text.

Limit sets can be renamed or removed by first clicking on the appropriate limit set name and then clicking the ***Rename*** or ***Remove*** buttons respectively. The column order of the limits sets can be changed by clicking on a limit set name and dragging the column to a new position in the grid. The “Default” limit set cannot be renamed, moved, or removed.

Limit set data can also be saved to a file by clicking the ***Save to file*** button. A standard FileSaveAs dialog box will appear. The initial directory will be set to the LimitSets subdirectory in the main Cell Assistant directory. All limit set data will be saved in a comma separated variable (*.csv) format.

Click the ***OK*** button to save all changes and return to the Limits page. Click the ***Apply*** button to save all changes so far and remain on this page. Click ***Cancel*** to return to the Limits page; all changes since the last time the ***Apply*** button was pressed (if any) will be lost.

CurrentLimitSet Property

When a TestPlan is started, the “Default” limit set will be active initially. The currently selected limit set can be changed using the following line of DSBasic code.

```
CurrentLimitSet = “NewLimitSetName”
```

Cell Assistant will instantly switch to this new set of limit setpoint values. The “Default” set can be reactivated at any time as follows.

```
CurrentLimitSet = “Default”
```

A runtime error will occur if the specified limit set name is not valid.

Note that other limit properties depend on the currently selected limit set. For example, retrieving and/or setting the High and Low setpoint values (using the previously described DSBasic code) always operate on the currently selected limit set.

The name of the currently selected limit set can be retrieved by a running TestPlan as follows:

```
StringValue = CurrentLimitSet
```

Chapter 7 - TestPlans

In order for the computer to run a test, it must execute a set of properly written instructions. We call this set of instructions a TestPlan. A TestPlan is written in a language called DSBasic.

About DSBasic

DSBasic (short for Dyne Systems Basic) is the programming language used to create the instructions which control a test run by the computer. DSBasic is modeled after Microsoft's Visual Basic for Applications. It shares many similarities to Microsoft's Visual Basic but has some important differences.

One major difference is that Visual Basic is an object oriented, event driven language. DSBasic is object oriented, but it is not event driven. We say that DSBasic is a procedure driven language. This was chosen because most tests run in a sequence. They generally have a beginning, a middle, and an end; and flow in a consistent order. DSBasic is also configured to run this way and so is easier to use with these applications.

TestPlan Files window

This window contains a list of the TestPlans that are written and contained in the present TestSystem.

TestPlan Notes window

This window contains notes that pertain to the highlighted TestPlan. To add or edit a note, simply position the cursor within the window and click. You will now be able to add new text or edit existing text.

Add button

To use this, you must be logged on in the configuration level. To add a new TestPlan, click the ***Add*** button. This brings up the ***Add TestPlan*** dialog box. Fill in the desired name in the ***TestPlan Name*** text box and click ***OK***. This will bring up the TestPlan editor. For more information about the TestPlan editor and writing DSBasic TestPlans, see the DSBasic User's Guide.

Modify button

To use this, you must be logged on in the configuration level. Once a TestPlan has been added, it can be edited. To do this, select the TestPlan that you wish to modify and click the ***Modify*** button. This will bring up the TestPlan editor. For more information about the TestPlan editor and writing DSBasic TestPlans, see the DSBasic User's Guide.

Rename button

To use this, you must be logged on in the configuration level. To rename a TestPlan, select the TestPlan that you wish to rename and click the ***Rename*** button. This will bring up the TestPlan Rename dialog box. Enter the new name of the TestPlan and click the ***OK*** button.

Remove button

To use this, you must be logged on in the configuration level. To remove a TestPlan, select the TestPlan that you wish to rename and click the ***Remove*** button. A warning message box will be displayed which gives you one chance to change your mind. Click the ***OK*** button to remove the selected TestPlan; otherwise, click the ***Cancel*** button.

Run Button

To run a TestPlan, select the TestPlan that you wish to run by clicking it. Once it is highlighted, click the ***Run*** button. The TestPlan will then begin to run. If you wish to stop the TestPlan from running, click on ***Control*** on the main menu then click ***TestPlan Stop*** or ***Stop Data Acquisition***. Be sure to bring any rotating equipment to a stop manually or press the hardware Emergency Stop button.

Set Default button

The default TestPlan is identified for several reasons. First, when a TestSystem is opened and the Play button on the Tool Bar is clicked (green arrow pointing to the right), the default TestPlan will run. This eliminates the step of picking the TestPlan from the list. Secondly, when limits or calculated channels are configured, the DSBasic code can be written into the default or global TestPlan; thus, make sure the desired TestPlan is selected as the default TestPlan if you are creating limits and or calculated channels and choosing the default TestPlan as the location to place their implementation. To set the default TestPlan, select the desired TestPlan and click the ***Set Default*** button. This will cause the highlighted TestPlan to be designated the Default TestPlan. The TestPlan name will then appear above the ***TestPlan Files*** window next to the ***Default TestPlan*** text.

Edit Global button

To use this, you must be logged on in the configuration level. To edit the Global TestPlan, click the ***Edit Global*** button. This will bring up the TestPlan editor. For more information about the TestPlan editor and writing DSBasic TestPlans, see the DSBasic User's Guide.

The Global TestPlan was created in order to have a place to store DSBasic code that can be used by any or all of the TestPlan files listed in the ***TestPlan Files*** list box. The DSBasic code in each TestPlan file can only access functions, subroutines, and variables that exist within that same TestPlan file; i.e., they cannot access functions, subroutines, or variables that have been declared in any other TestPlan file. The exception to this rule is the Global TestPlan. Each TestPlan file can access its own functions, subroutines, and variables AND those in the Global TestPlan. This makes the Global TestPlan an ideal place to store functions, subroutines, etc... that are needed by more than one TestPlan file. Any changes that are made in the Global TestPlan (e.g. design changes or code fixes) automatically affect all other TestPlan files.

Limit procedures and TestPlan calculated functions are two examples of DSBasic code that should probably be placed in the Global TestPlan; both have been described previously in this document. For example, all running TestPlans probably want the same implementation of a TestPlan calculated function, so it makes no sense to replicate a `CLC_Whatever` function in every TestPlan; instead, put it in the Global TestPlan where all TestPlan files can use it.

All Cell Assistant objects (i.e. Limits, Channels, Data Files, etc...) can also be accessed by the Global TestPlan.

Since multiple TestPlans can be edited at the same time, the title bar of each edit window displays the name of the TestPlan being edited in that window. When editing the Global TestPlan, the title bar reads ***TestPlan Modify: GLOBAL_TESTPLAN***. In addition, the ***Start***, ***Break***, and ***End*** buttons of the edit window tool bar are disabled when editing the Global TestPlan. This is because the Global TestPlan only contains commonly used subroutines; it NEVER contains the start of a test (i.e. a "Main" subroutine). Debugging always starts in one of the other TestPlan files. You can still "step into" a subroutine in the Global TestPlan from a subroutine in another TestPlan file. Once in a Global TestPlan subroutine, you can continue to "step" through your code by clicking the ***Single Step*** button (which is not disabled).

Several interesting situations can occur that you should be aware of as you implement various TestPlans that also use the Global TestPlan. Confusion will be avoided by understanding what the DSBasic compiler does every time a TestPlan is compiled (i.e. the ***Run*** button is pressed). When any TestPlan is compiled, the Global TestPlan is also compiled along with it. TestPlan references to functions, procedures, variables, etc ... are resolved by FIRST looking in the TestPlan itself. If not found, the Global TestPlan will be checked. If the reference is still not resolved, a compiler error is generated. Knowing this, here is a list of programming idiosyncrasies to be aware of.

- It is possible to have a subroutine (or function) in the compiled TestPlan that has the same name as a subroutine in the Global TestPlan. In this case, the subroutine in the TestPlan will be used and the conflicting subroutine in the Global TestPlan will be completely ignored. This can be done by design or by accident. Some TestPlans may want to "override" a subroutine in the Global TestPlan by providing their own implementation that is slightly different. On the other hand, this situation can also occur accidentally if a TestPlan writer creates a new subroutine in a TestPlan that just happens to match the name of a subroutine in the Global TestPlan. Cell Assistant issues no warnings if this occurs. This problem can be avoided by applying good software design and organization procedures.
- If a "Main" subroutine is placed in the Global TestPlan, it will never be called. Each TestPlan contains a "Main" routine that serves as the entry point to start execution of that TestPlan. The Global TestPlan merely contains subroutines, functions, etc ... that are commonly used by other TestPlans. The Global TestPlan is not really a TestPlan in and of itself and, thus, does not need a "Main" subroutine.
- If a subroutine in the Global TestPlan references subroutines implemented in any other TestPlan, then ALL of the TestPlans must implement these subroutines and variables. If not, compiler errors will be generated. This is easy to understand by recalling that whenever any TestPlan is compiled, the Global TestPlan is compiled along with it. All external subroutine, function, and variable references in the Global TestPlan must be resolved every time the Global TestPlan is compiled. And since the Global TestPlan is compiled when every TestPlan is compiled, every TestPlan must provide the necessary subroutines, etc ... to resolve the references by the Global TestPlan.

Properties button

To use this, you must be logged on in the configuration level. Certain properties exist that can be uniquely configured for each listed TestPlan file (except the Global TestPlan). To change any of these properties for a given TestPlan, select this TestPlan and click the ***Properties*** button. A dialog box will be displayed that lists all of the current settings of the TestPlan properties for the selected TestPlan.

Flight Recorder Properties

Set the ***Log Rate and Duration*** by clicking the down arrow at the right of the drop-down list box. Your choices are to record data at ***1.0 Hz for 10 minutes*** or to record data at ***0.1 Hz for 100 minutes***. The flight recorder feature is explained in the following paragraphs. This should help you to choose the best property settings for your application.

The Flight Recorder is a background task in Cell Assistant that automatically logs important test information (e.g. channel data values, limit trips, etc...) to a data file whenever a TestPlan is running. These data files are referred to as Flight Recorder files. They are mainly used to analyze TestPlans that have shutdown and/or terminated prematurely due to errors. When a TestPlan stops (for any reason), the Flight Recorder file will contain a history of all channel data values and other events that occurred for a specified time interval prior to the stopping of the TestPlan. They can be viewed with a standard spreadsheet program. The Flight Recorder ***Log Rate and Duration*** property setting explained above requires you to make a tradeoff decision between the amount of history you wish to retain prior to a TestPlan stop versus the rate at which all channel data is saved.

The format of a Flight Recorder file is the same as the format of “comma separated variable” (*.csv) Data Files described in Chapter 8. Unlike Data Files, the contents of Flight Recorder files cannot be configured. The Flight Recorder files contain snapshots of ALL defined TestSystem channels taken at the specified logging rate. Also, whereas Data Files can be opened, closed, started, and stopped under the complete control of a running TestPlan, Flight Recorder data logging starts when a TestPlan starts and stops when the TestPlan stops. Note that in almost all debugging situations, regular TestPlan Data Files (as described in Chapter 8) are a better choice because they can be configured to log only the channels of interest, the logging rate can be increased to 20 Hz, and logging can be selectively started and stopped under complete control of the running TestPlan. The Flight Recorder task is designed to be a “backup mechanism” in situations where Data File are not (or cannot) be used.

All Flight Recorder files are saved in the “FlightRecorder” subdirectory of the main Cell Assistant subdirectory. Typically this would be “C:\CA4W\FlightRecorder”. The Flight Recorder filenames are of the form “FlightRecXX.csv” where XX ranges from 00 to 99. When you run your first TestPlan, the file FlightRec00.csv will be created. When the second TestPlan is run, the file FlightRec01.csv fill be created. When the 100th TestPlan is run, Flightec99.csv will be created. Finally, when the 101th TestPlan is run, a new FlightRec00.csv will be created and will replace the first one. This sequence continues forever. When viewing the files in the “FlightRecorder” subdirectory, always sort the file list by “creation date” or “last modified date” in order to determine which flight recorder file was created last (and hence is the one from the most previous TestPlan run).

TestPlan Programming Topics

This section contains various programming topics that have not been covered elsewhere in this manual.

Cell Assistant extensions to DSBasic

Other chapters in this manual describe how to configure channels, limits, data file formats, and VIP’s. These chapters also describe how to access and/or reference channels, limits, data file formats, and VIP’s in a running TestPlan (which is implemented in the DSBasic programming language). The ability to access these items in the DSBasic language is accomplished via language extensions. This is explained in more detail below.

DSBasic is already an object oriented language. You can programmatically create objects of various kinds and manipulate an objects properties and/or invoke any of the objects methods. Each time a TestPlan is run, Cell Assistant automatically creates are large set of global objects that can be accessed by the TestPlan. One channel object is created for every virtual channel that has been created and configured in the ***Channels*** page. Similarly, one limit object, one data file object, and one VIP object is created for every limit, data file format, and VIP that has been created and configured on the respective pages.

These global objects act like any other object in the DSBasic language except that these objects provide an interface between the running TestPlan and underlying Cell Assistant data acquisition subsystem. Setting an object to a value (i.e. object.value = N) typically just sets a value in some memory location in the computer. Setting the value of a channel object, on the other hand, does more than that. Setting the value of a channel object that happens to be an output channel will cause a command of some type to be sent to a device. Similarly, accessing the value of a channel object actually returns a data value that was sampled from a device by the data acquisition subsystem.

This concept needs to be clearly understood. When accessing Cell Assistant objects in a TestPlan, you may actually be communicating with an external device.

Object References

There are two methods by which any object can be accessed by a running TestPlan. The first method is discussed in the respective chapters on channels, limits, data file formats, and VIP's. This is the direct method where an object is accessed using its name. For example, every channel object has the same name as the channel it represents. For example, a channel named "aocDynoSetpoint" will be represented in the TestPlan by a channel object named "aocDynoSetpoint". You can then send a setpoint value to the Dynamometer controller in the TestPlan using the following line of DSBasic code.

```
aocDynoSetpoint = 100
```

Cell Assistant objects can also be accessed via a "reference" to the object. In DSBasic, you can create an object that can be used to reference other objects of the same type. The reference object is NOT an object itself. It must reference another object in order to be used. An example is shown below.

```
Dim aocReferenceToAnyAnalogChannel As AnalogChannel
Set aocReferenceToAnyAnalogChannel = aocDynoSetpoint
aocReferenceToAnyAnalogChannel = 100
```

This example accomplishes the same thing at the previous example. The first line creates (i.e. "Dim"s) the reference object. The second line establishes the channel object that will be referenced. Note that the reference object can only reference objects of similar type. In this example, "aocDynoSetpoint" must be an analog channel. Finally, in the third line, the value of the reference object is set which actually sets the value of the "aocDynoSetpoint" channel. Similarly, any other method/property of the original object can be invoked/modified using the reference object instead.

This example does not really demonstrate the true usefulness of using object references. It is most useful when the same set of operations has to be performed on several objects. It is easier to write a subroutine that performs these operations on an object reference. The "Set" command can then be used to set the reference object to the appropriate object prior to calling your subroutine that operates

on the reference object. Object references are more likely to be used with limit objects and data file format objects than channel objects.

A list of all Cell Assistant objects that can be “Dim”ed in order to create reference objects is shown below.

AnalogChannel	Limit
AverageChannel	DataFileFormat
CounterChannel	VIP
DDEChannel	
DigitalChannel	
TextChannel	
TimerChannel	

Electronic Mail (i.e. E-Mail)

The DSBasic language has been extended to include e-mail capability. Simple mail messages can be sent to a single person or a group of persons. A file can be optionally attached to any e-mail message. E-mail functionality must be enabled in the ***Preferences*** section.

A simple e-mail message consists of a sending address, a subject line, and a message line. A file can optionally be attached. The following TestPlan code can be used to send a simple message (with no attached file).

```
Email.Address = “Joe Blow”
Email.SubjectLine = “Cell Assistant Test Status”
Email.MessageLine = “Test completed: no errors”
Email.SendMail
```

Similarly, the following TestPlan code can be used to send a simple message that also contains an attached file.

```
Email.Address = “Joe Blow”
Email.SubjectLine = “Cell Assistant Test Status”
Email.MessageLine = “Test completed.  See attached file.”
Email.AttacheFile = “D:\\Ca4w\\DataFiles\\TestData001.csv”
Email.SendMailWithFile
```

The e-mail address can only specify a single name. This name can be the name of a single user or a name of a mailing list.

Chapter 8 - Data Files

Probably the most important function of Cell Assistant is the recording of data. Data recording is configured by clicking the ***Data File*** tab in the TestSystem property sheet. All data is stored into a file. The way the files are configured controls everything about how data gets stored. To configure data files, you must be logged in at the configuration level.

Adding Data File Formats

To add a data file format, click the ***Add*** button. This brings up the ***Add*** dialog box.
Type in the desired name in the ***Data File Format*** text box and click ***OK***.

Renaming Data File Formats

To rename a data file format, click the ***Rename*** button. This brings up the ***Rename Data File Format*** dialog box. Type in the new name in the ***Data File Format*** text box and click ***OK***. Note that all references to this Data File Format in any and all TestPlans must also be updated.

Removing Data File Formats

To remove a data file format, click the ***Remove*** button. This brings up a confirmation message box that gives you one chance to change your mind. To go ahead and remove the selected data file format, click ***OK***; otherwise, click the ***Cancel*** button.

Configuring Data File Formats

To configure a data file format, highlight the data file format (in the ***Defined Data File Formats*** list box) that you wish to configure. Click the ***Modify*** button. This brings up the ***Modify*** dialog box which allows you to select which channels are recorded by that data file format and their order. On the left side of the dialog is the ***Available Channels*** list. On the right is the ***Selected Channels*** list. To cause a channel to be recorded in the data file, highlight the desired channel in the ***Available Channels*** list and click the ***Add*** button. The channel will now move from the ***Available Channels*** list to the ***Selected Channels*** list. Continue this process until all of the desired channels are shown. Every channel available can be moved to the ***Selected Channels*** list by clicking the ***Add All*** button. If you wish to remove a channel from the ***Selected Channels*** list, highlight the desired channel in the ***Selected Channels*** list and click the ***Remove*** button. The channel will now move from the ***Selected Channels*** list back to the ***Available Channels*** list. Every channel selected can be removed from the ***Selected Channels*** list by clicking the ***Remove All*** button.

The order in which the selected channels are stored in the data file can also be altered. You may have noticed that if a channel in the ***Selected Channels*** list is highlighted, adding a channel from the ***Available Channels*** list causes the channel to be added above the highlighted channel. By using this feature, you can determine the order in which the data is recorded.

Exit the configuration dialog by clicking the ***OK*** button.

To determine the type of file the data file will be, click the down arrow at the right of the ***Format*** drop-down list box in the ***File*** group box. Choosing “Comma Separated Variable” will create a *.csv

file that is easily imported into a spreadsheet. Choosing “Flat ASCII Text” will create a *.txt file that is easily read with a word processor.

To determine the rate that the channels will be logged to disk, click the down arrow at the right of the ***Log Rate*** drop-down list box in the ***File*** group box. The number shown represents the number of times per second (Hertz) that the data will be recorded to disk. Please note that this rate can be changed from the TestPlan.

If the ***Include Limit Exceptions*** check box is left empty, no message will be recorded when a limit is tripped. Check this box if you want messages to be recorded to the data file when limits are tripped. If this check box is checked, any time a limit is tripped, Cell Assistant automatically writes a line of limit data (identified by “LM” in the first column). This line documents the name of the limit that tripped, the channel that the limit was monitoring, the type of limit (high, low, etc ...), and the high and/or low limit settings (depending of the limit type)..

Using Data Files in TestPlans

A data file is made up of rows or lines of data. The first piece of information in a data line always shows the line type. Typically each row contains the values of the desired channels, however, there are several types of data lines. These include header lines (designated by “HD” in the first position), forced entry lines (designated by “FE” in the first position), time logged lines (designated by “TL” in the first position), snapshot lines (designated by “SL” in the first position), and limit lines (designated by “LM” in the first position). These will be explained later in detail.

The second piece of information in a data line is always the date that the line was recorded, and the third piece is always the time that the line was recorded. Subsequent information depends on the type of line it is.

The following sections describe how to control how Cell Assistant records data.

Open Method

The first thing that has to be done in a TestPlan when you wish to log data is to open the data file and associate a data file name with the proper data file format. It is important to realize that many different data files, with different file names, can be created based on a single data file format; however, these different data files cannot be created at the same time using a single data file format. In other words, once you use the Open method on a data file format, subsequent attempts to use the Open method will fail until the Close method is called Also, more than one data file can be written to during a test (as long as you use different data file formats as just explained). The Open method takes the form:

```
FormatName.Open “Filename” [, Mode, [NoHeaderFlag]]
```

where FormatName is the name of the data file format. **Filename** is the actual name of the file that’s being created. Note that “**Filename**” is a string and cannot contain a path or extension. **Mode** determines whether the file will be added (i.e. appended) to or overwritten. If added to, data will be added to the end of the file. The allowed **Mode** values are “Add” or “Overwrite”. **Mode** is an optional parameter. If omitted, the default mode will be to add to the file. The following are all valid Open methods:

```
Format1.Open “MyDataFile”, Add

Format1.Open “MyDataFile”, Overwrite
```

```
Format1.Open "MyDataFile"     (defaults to Add mode)
```

When the Open method is executed, Cell Assistant automatically writes three lines of header data (identified by "HD" in the first column). The first line documents when the file was opened and what format was used. The second line documents the channel names. The third line documents the units that the channel values are in.

The optional **NoHeaderFlag** parameter allows a data file to be opened without adding the three header lines to the data file. This allows you to create "numbers only" data files that may be easier to analyze later on. To prevent the logging of the header lines, set the **NoHeaderFlag** to **NoHeader** or **NH**. If this parameter is omitted, the default action is to always log the header lines. An example is shown below.

```
Format1.Open "MyDataFile", Add, NH
```

If desired, the second and third header lines (i.e. the channel names and channel units respectively) can be written to the data file more than once. See the sections on the WriteChannelNames, WriteChannelUnits, and WriteChannelHeader methods. Writing the channel names and/or channel units at other places in the data file can often improve the readability of the data file later on.

If you look on the disk at the end of the test (or after the file is closed - see the Close Method section), you will find the data file named "MyDataFile.csv" or "MyDataFile.txt" (depending on the format) in the "DataFiles" subdirectory of the main Cell Assistant subdirectory. Typically this would be C:\CA4W\DataFiles\Filename.CSV or C:\CA4W\DataFiles\Filename.TXT.

StartLogging Method

This method causes the system to start logging data to the file at the specified rate. The Open method must first be executed for this method to function. The method takes the form:

```
FormatName.StartLogging [Rate[, NoHeaderFlag]]
```

where **Rate** is optional and can be 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10, or 20. The number is in samples per second or Hertz (Hz). The following are valid StartLogging commands:

```
Format1.StartLogging

Format1.StartLogging 0.01

Format1.StartLogging 10
```

When the StartLogging method is executed, Cell Assistant automatically writes a line of header data (identified by "HD" in the first column). This line documents when logging was started and at what rate. After that, at the specified logging rate, Cell Assistant automatically writes lines of data which contains the values of the specified channels. These data lines are identified by "TL" in the first column.

The optional **NoHeaderFlag** parameter allows data logging to be started without adding the header line to the data file. This allows you to create "numbers only" data files that may be easier to analyze later. To prevent the logging of the header line, set the **NoHeaderFlag** to **NoHeader** or **NH**. If this parameter is omitted, the default action is to always log the header line. An example is shown below.

```
Format1.StartLogging 10, NH
```

LogRate Property

This command changes the rate at which data is written to the data file. The property takes the form:

```
FormatName.LogRate = Rate
```

where **Rate** is the rate at which logging will occur and can be 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10 or 20. The number is in samples per second or Hertz (Hz). The following is a valid command:

```
Format1.LogRate = 1.0
```

StopLogging Method

This method causes the system to stop logging data to the file. The Open method and the StartLogging method must first be executed for this method to function. The method takes the form:

```
FormatName.StopLogging [NoHeaderFlag]
```

The following is a valid command:

```
Format1.StopLogging
```

When the StopLogging method is executed, Cell Assistant automatically writes a line of header data (identified by "HD" in the first column). This line documents when logging was stopped and at what rate it was being logged.

The optional **NoHeaderFlag** parameter allows data logging to be stopped without adding the header line to the data file. This allows you to create "numbers only" data files that may be easier to analyze later on. To prevent the logging of the header line, set the **NoHeaderFlag** to **NoHeader** or **NH**. If this parameter is omitted, the default action is to always log the header line. An example is shown below.

```
Format1.StopLogging NH
```

SnapShot Method

This method writes one sample of data to the data file. The Open method must first be executed for this method to function. The method takes the form:

```
FormatName.SnapShot
```

The following is a valid command:

```
Format1.SnapShot
```

When the SnapShot method is executed, Cell Assistant automatically writes one line of snapshot data which contains the values of the specified channels. Snapshot entry data lines are identified by "SL" in the first column.

ForceEntry Method

This method writes a specified string to the data file. The Open method must first be executed for this method to function. The method takes the form:

```
FormatName.ForceEntry "Your string here"
```

The following is a valid command:

```
Format1.ForceEntry "Engine Serial Number is 12345"
```

When the ForceEntry method is executed, Cell Assistant automatically writes one line of data which contains the specified string. Forced entry data lines are identified by “FE” in the first column.

Close Method

This method closes a data file. The Open method must first be executed for this method to function. The method takes the form:

```
FormatName.Close [NoHeaderFlag]
```

The following is a valid command:

```
Format1.Close
```

When the Close method is executed, Cell Assistant automatically writes one line of header data (identified by “HD” in the first column). This line records the format of the data file being closed and the time and date.

The optional **NoHeaderFlag** parameter allows the data file to be closed without adding the header line to the data file. This allows you to create “numbers only” data files that may be easier to analyze later on. To prevent the logging of the header line, set the **NoHeaderFlag** to **NoHeader** or **NH**. If this parameter is omitted, the default action is to always log the header line. An example is shown below.

```
Format1.Close NH
```

IsOpen Method

This method allows a TestPlan to check if a data file is currently open using this data file format. If so, this method returns TRUE. If not, this method returns FALSE. This method has the following form:

```
FormatName.IsOpen
```

The following is a valid usage of this method. It is typically used in shutdown and error handling routines that perform cleanup after and error. The TestPlan can check if a data file is actually open and, if so, close it.

```
If Format1.IsOpen Then
      Format1.Close
End If
```

WriteChannelNames Method

This method writes all channel names to the data file. The Open method must first be executed for this method to function. The method takes the form:

```
FormatName.WriteChannelNames
```

The format of the channel names is the same used in the header entry made by the Open method. This method can be used to periodically rewrite the channel names to the data file in order to improve the readability of the data file.

WriteChannelUnits Method

This method writes all channel units to the data file. The Open method must first be executed for this method to function. The method takes the form:

```
FormatName.WriteChannelUnits
```

The format of the channel units is the same used in the header entry made by the Open method. This method can be used to periodically rewrite the channel units to the data file in order to improve the readability of the data file.

WriteChannelHeader Method

This method writes all channel names and units to the data file. The Open method must first be executed for this method to function. The method takes the form:

```
FormatName.WriteChannelHeader
```

This method is equivalent to calling the WriteChannelNames and WriteChannelUnits methods except this method guarantees that the channel names and channel units entries are logged in consecutive rows in the data file.

Copy Method

This method stops data logging (if necessary), closes the data file, and makes a copy of the data file. Afterwards, the file is reopened and data logging is restarted (if necessary). The method takes the form:

```
FormatName.Copy [NoHeaderFlag]
```

This method is used to create copies of a data file to be used by other applications that wish to view intermediate results of a running TestPlan. The copied file will reside in the same directory as the original data file. It will have the same file name with a "Copy" suffix. For example, if the currently open data file is named "xxx.csv", the Copy method will create the file "xxxCopy.csv". If this file already exists, it will be overwritten. An entry is made in the original data file noting that a copy has been made. When the TestPlan stops, all files that were created using this Copy method will automatically be deleted.

The optional **NoHeaderFlag** parameter allows the data file to be copied without adding the header line to the data file. This allows you to create "numbers only" data files that may be easier to analyze later on. To prevent the logging of the header line, set the **NoHeaderFlag** to **NoHeader** or **NH**. If this parameter is omitted, the default action is to always log the header line. An example is shown below.

```
Format1.Copy NH
```

FileUpdate Method

This method forces the file system to write all currently logged data to the system disk. The method takes the form:

```
FormatName.FileUpdate
```

During normal operation, all data that is logged to a data file initially goes into memory buffers that reside in the file system drivers. The file system drivers then write the data to disk as the buffers become full. This happens transparently and greatly improves the real-time performance of the data logging software. Also, the file system doesn't update the directory until the data file is closed by the running TestPlan.; thus, an external application will not "see" a new file or the additions to an existing file until the data file has been closed.

This method can overcome this limitation. The FileUpdate method forces all currently buffered data to be written to disk and the disk directory is updated with the new file size. Other applications now

have read-only access to the current contents of the data file. This method is used to make intermediate data file data available to other applications.

Sharing Data Files with other applications

A running TestPlan typically creates data files for analysis that is performed at a later time (i.e. after the TestPlan has stopped). It is possible, however, to access these data files "on-the-fly" while the TestPlan is running. This feature was explained somewhat in the previous section on the Data File Format **FileUpdate** method. It is summarized again here. A TestPlan needs to be written correctly in order to make this feature useful; on the other hand, using this feature also has some real-time drawbacks that you should be aware of.

All data files are created with READ share permission to other applications; thus, a spreadsheet program could access a data file even while a TestPlan is running and has the data file open for logging. Only READ sharing is allowed, so the spreadsheet program will be unable to modify and/or or save any changes to the disk file. The TestPlan must use the **FileUpdate** method to make this feature useful. When a data file is initially created, it will NOT appear in the system disk directory until the data file is closed by the TestPlan. A file of size = 0 bytes will initially appear in the directory listing. When the TestPlan executes the **FileUpdate** method, Cell Assistant will force the file system to write all data logged so far to the system disk and update the directory. This makes all data that has been logged so far available to other applications. Similarly, a data file that is being added to will not appear to have grown in size (by looking at the disk directory) until the file has been closed or the **FileUpdate** method is executed. Clearly, the FileUpdate method must be periodically called in order to "flush" all logged data to the system disk.

This feature can negatively impact the real-time performance of data logging. First of all, periodically flushing data to the disk using the **FileUpdate** method disturbs the real-time efficiency of the file system driver software. The file system typically caches (i.e. buffers) all logged data in internal memory buffers and only writes data to the disk as the buffers become full. Excessive flushing destroys the effectiveness of this design. Second of all (and probably most important), some logged data will probably be lost at the exact instant that another application reads the data file. Even though other applications only require READ access to the data file, the file system will still block all access to the data file during the "instant" that the data file is being read in. This "instant" can actually last for several seconds for large data files and during this "instant", Cell Assistant is also denied access to the data file. For large channel counts and/or high data logging rates, the internal memory buffers will not be able to buffer all logged data for this duration; thus, some data will be lost. Cell Assistant will resume data logging as soon as access to the data file is granted. Two SYSTEM device channels have been defined that indicate the occurrence of data overruns. See Appendix A for more information about these channels. They should be included in all data files that are to be shared by other applications such that data overruns are clearly marked in the data file.

Chapter 9 - Calibration

Calibration is done by clicking the ***Calibration*** tab in the TestSystem property sheet. This exposes the ***Calibration*** property sheet. To use this, you must be logged on in the calibration or configuration level.

Calibration allows the user to expose a transducer to a known reference value and adjust the instrumentation system to read the same value as the known reference value. For example, a system is measuring oil pressure through a 100 psi pressure transducer. To verify proper system operation, the user connects a pump and a very accurate pressure gage to the pressure transducer inlet. The pump is then pumped until the gage reads exactly 80.00 psi (generally this is accomplished with a pressure calibrator similar to those built by Druck, Inc.) . Cell Assistant data acquisition is started, system diagnostics are viewed, and the operator views the value that Cell Assistant is reading, knowing that Cell Assistant *should* be reading 80.00 psi. If Cell Assistant is reading close enough (it's up to the operator to decide how close is "close enough") then no further action is needed. If, however, it is not "close enough", then that channel needs calibration.

To calibrate a channel, Cell Assistant needs to compute a gain correction value and an offset correction value. Once these values are computed, Cell Assistant takes the uncalibrated value of the channel, multiplies it by the gain correction value, then adds the offset correction value to the product. The formula is shown below:

Calibrated Value = (Uncalibrated Value * Gain Correction) + Offset Correction

Cell Assistant is designed with the assumption that the Calibrated Value and the Uncalibrated Value are going to be pretty close to each other (within 30%). This way, the gain correction for each channel should be pretty close to 1. If a channel is drifting for some reason, the fact that the gain correction drifts farther and farther from 1 should be easy to see. A warning is displayed if the gain correction is not between 0.7 and 1.3.

To calibrate a channel, highlight the desired channel in the ***Channels*** list box and click the ***Calibrate*** button. This will bring up the ***Calibrate Input Channel*** dialog box. The next step is to select which method of calibration to use - 1 Point, 2 Point, or Manual. The three methods are described below.

1 Point Calibration

1 Point calibration allows the channel to be calibrated only at a lower input value. It only effects the offset correction.

To calibrate using the 1 point method, select the 1 Point option button. Notice that only the ***Low Point*** text boxes are enabled. Click the ***Activate*** button. This will cause Cell Assistant to start acquiring data. The actual live data will be displayed in the ***Uncalibrated Low Point*** text box. Now expose the transducer to some known value near the bottom of the desired range. This will typically be zero but it doesn't have to be. Next type in the desired reading in the ***Desired Low Point*** text box. It's a good idea to make sure that Uncalibrated and Desired readings are close. If they are not, something may be wrong or the slope and offset for conversion to engineering units may be wrong.

2 Point Calibration

2 Point calibration allows the channel to be calibrated at both an upper input value and a lower input value. It effects both the gain correction and the offset correction.

To calibrate using the 2 point method, select the 2 Point option button. Notice that both the ***High Point*** and ***Low Point*** text boxes are enabled. Click the ***Activate*** button for either the high point or the low point (for this example we'll assume that we started with the low point). This will cause Cell Assistant to start acquiring data. The actual live data will be displayed in the ***Uncalibrated Low Point*** text box. Now expose the transducer to some known value near the bottom of the desired range. This will typically be zero but it doesn't have to be. Next type in the desired reading in the ***Desired Low Point*** text box. It's a good idea to make sure that Uncalibrated and Desired readings are close. If they are not, something may be wrong or the slope and offset for conversion to engineering units may be wrong. When the readings are acceptable, click the ***Accept*** button.

Click the ***Activate*** button for either the High Point. The actual live data will be displayed in the ***Uncalibrated High Point*** text box. Now expose the transducer to some known value near the top of the desired range. Type in the desired reading in the ***Desired High Point*** text box. It's a good idea to make sure that Uncalibrated and Desired readings are close. If they are not, something may be wrong or the slope and offset for conversion to engineering units may be wrong. When the readings are acceptable, click the ***Accept*** button.

Manual Calibration

This method is for use when the needed gain and offset correction factors are known. Simply type the desired values into the ***Gain Correction*** and ***Offset Correction*** text boxes.

Once the calibration has been done, the resulting Gain and Offset Correction factors are displayed in their respective text boxes. Also displayed is the change in the gain and offset correction factors from the last calibration. This is included to show if a transducer is drifting farther and farther away from correct calibration which may indicate a failing transducer.

There is a comment box at the bottom of the dialog box. Enter any additional information needed such as the serial number of the calibrator used, the date that the calibrator was calibrated, and any other information that may be needed. The time and date of the calibration is automatically recorded, as is the method of calibration and the calibration values used.

Click the ***OK*** button.

Saving Calibrations

Calibration information used by Cell Assistant is stored in the Windows NT registry. Once the system has been calibrated, the gain and offset correction factors along with additional information can also be saved to a file for storage, to recalibrate the system later, or for use in recording a calibration history. To do this, click on the ***Save*** button. This will launch the ***Save Calibrations*** dialog box. Cell Assistant will generate a sequentially numbered calibration file. This is only a suggestion. If a different file name is desired, enter it in the ***File name:*** text box. Click the ***Save*** button to write the file.

Calibration File Characteristics

Calibrations are saved in an ASCII comma separated variable format (.csv file extension) like data files. Each comma in a line separates a column of data. The information held in the columns is as follows:

Column A	Subchannel
Column B	Source device
Column C	Channel name
Column D	Name of TestSystem that the calibration was done in
Column E	Date/Time stamp
Column F	Method of calibration
Column G	Actual low point (the uncalibrated reading of the channel)
Column H	Desired low point (what the channel will read after calibration)
Column I	Actual high point (the uncalibrated reading of the channel)
Column J	Desired high point (what the channel will read after calibration)
Column K	Gain correction factor
Column L	Offset correction factor
Column M	Comment

Restoring Calibrations

If required, the system can be recalibrated based on a calibration file saved earlier. Click the ***Restore*** button on the ***Calibration*** property page. Choose the desired calibration file from the list and click the ***Open*** button. The system calibration information from the selected file will be copied into Cell Assistant. This replaces the current calibration data so make sure you save the current calibration data to a file (if you haven't done so already).

Chapter 10 - Virtual Instrument Panels

Virtual Instrument Panels - or VIP's - are on-screen displays of data being acquired by Cell Assistant. VIP's consist of meters, gages, buttons, pilot lights, and a variety of other controls that allow a user to interact with a running TestPlan in Cell Assistant. VIP's are actually executable programs that are developed using Microsoft Visual Basic (which is generally included with the Cell Assistant Software). Once they are created, they are compiled using Visual Basic into a standard executable file format (*.exe file) and placed in the VIP subdirectory of the main Cell Assistant directory.

This chapter consists of three main sections. The first section describes how Cell Assistant interfaces with a VIP once it has been created, compiled, and placed in the \CA4W\VIP subdirectory. The second section describes how to actually create the VIP using Microsoft Visual Basic. The third section describes the DyneSystemsCo. VIP Controls ActiveX control set that is used to create high quality user-interfaces.

In order to configure VIP's click the ***Virtual Instrument Panels*** tab on the TestSystem property sheet.

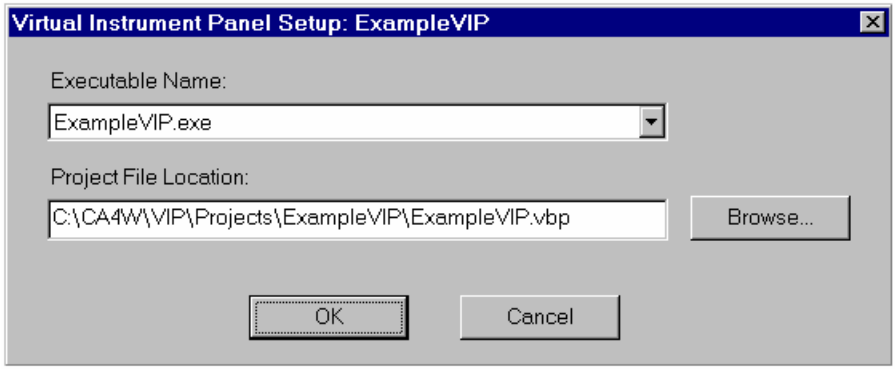
Using VIP's in Cell Assistant

The next few sections describe how to make an already created VIP available to Cell Assistant.

Adding VIP's

To add a new VIP to Cell Assistant, click the ***Add*** button. A dialog will be displayed that asks for the name of the new VIP. Cell Assistant will create a VIP object with this name every time a TestPlan is run. Click the ***OK*** button when done.

Another dialog will now appear that asks for the name of the executable file that implements this VIP. A drop-down list of all *.exe files in the \CA4W\VIP subdirectory is displayed. Select the appropriate *.exe file. This dialog also asks for the path to the Microsoft Visual Basic project file (*.vpb) that contains all of the source code that created the selected *.exe file. Enter the full path to this file or use the ***Browse*** button to locate it (which is typically easier). An example is shown below.



Cell Assistant needs to know the path to the project file since it will automatically launch Visual Basic for you and open the appropriate project when you click the ***Modify*** button. The path to where Microsoft Visual Basic has been installed needs to be set in the ***Preferences*** section of Cell Assistant.

The name of the *.exe file and the location of the project file for this VIP can be changed at any time by clicking the ***Setup*** button.

Renaming VIP's

To rename a VIP, click the ***Rename*** button and enter the new VIP name. This changes the name of the VIP object that is created when any TestPlan is run; thus, all references to this VIP in any TestPlan also need to be updated such that they reference the VIP using the new name.

This does NOT affect the name of the *.exe file or the project file.

Removing VIP's

A VIP can be removed from Cell Assistant by clicking the ***Remove*** button. This does NOT remove the *.exe file from the \CA4W\VIP subdirectory. It also does NOT remove any of the project files that generate the VIP.

Using VIP's in TestPlans

Using a VIP in a TestPlan is very simple. To display a VIP (in this case ExampleVIP) use the following code:

```
ExampleVIP.Launch
```

To kill a VIP that's running, use the following code:

```
ExampleVIP.Close
```

Creating VIP's

The next few sections describe how to create a Virtual Instrument panel using Microsoft Visual Basic and the DyneSystemsCo. VIPCONTROLS ActiveX control set. Microsoft Visual Basic allows you to create forms where you can place various gages, meters, knobs, etc … in order to create rich looking user-interfaces. No lines of code need to be written using the VIPCONTROLS control set.

Prior to the development of this control package, VIP's were created using control sets from other vendors and one special control from DyneSystemsCo. called the CADAQ control. This control merely provided access to Cell Assistant channel data. Visual Basic code had to be written to pass data to and from Cell Assistant and these other controls. The CADAQ control is described in Appendix D. It is no longer officially supported. You are encouraged to develop all new VIP's using the VIPCONTROLS.

Creating a new VIP project

In order to create a new VIP project, launch Microsoft Visual Basic from your desktop. Select ***File*** from the main menu, then select ***Save Project As***. Save the project using whatever name you wish. It should be descriptive of the VIP that you are creating. Note that you don't really need to select ***File*** and ***New Project*** because Visual Basic already starts with a new project when it is initially launched.

We recommend that you create a separate subdirectory for each VIP's project files, and that the project subdirectories be put under \CA4W\VIP\ProjectFiles subdirectory. For example, the following subdirectory would be created for the project files for the VIP called "ExampleVIP":

C:\CA4W\VIP\ProjectFiles\ExampleVIP

Each VIP consists of one or more forms. Each is saved as a form (*.frm) file. Once your forms are completely designed, they are compiled into an executable file. Each VIP also has an overall project (*.vbp) file.

Once you have added all controls to your forms and tested them, exit Visual Basic. Go back to Cell Assistant and add this VIP to your TestSystem as explained in a previous section.

Adding controls

All available controls are shown as icons in the Visual Basic toolbox. If Cell Assistant has been properly set up on your computer (See Appendix C), all of the icons for the DyneSystemsCo. VIPCONTROLS should appear in the toolbox. Simply click on any control that you want to add to your form and then drag it onto the form. The VIPCONTROLS require no programming at all. Just click on any control that is on the form and click the right mouse button. Then select ***Properties***. A set of property pages will now be displayed. The number and type of property pages depends on the selected control. Each VIPCONTROL can be entirely configured from these property pages.

The specific properties of each VIPCONTROLS control is described later on.

Debugging the VIP Code

Click the play button in the menu bar in Visual Basic. This will cause the code to run and the VIP to be displayed. If the VIP is not displayed, the debugger should point you in the right direction to start fixing the problem. Once the code runs correctly it can be compiled into an executable file.

Compiling the VIP Code

Click the ***File*** menu item in the menu bar in Visual Basic, then click the ***Make EXE File...*** menu item. Select a file name and enter it into the ***File Name:*** text box. Be sure that before you click on the ***OK*** button that the file will be located in the \CA4W\VIP subdirectory or Cell Assistant won't know where to find it. All executables compiled from projects should be kept in the \CA4W\VIP subdirectory because this is where Cell Assistant looks for them.

VIPCONTROLS ActiveX Control

The VIPCONTROLS ActiveX control provides a set of meters, gages, etc … that allows high-quality user-interfaces to be created with absolutely NO programming whatsoever. The controls can be placed on a Visual Basic form and then totally configured using standard property pages. Each of these controls is briefly described below. The details of each control property is not discussed since most have an obvious affect on the controls operation and/or appearance. On the other hand, some general concepts that apply to all controls will be presented.

AnalogMeter

The Analog Meter is a simple needle-type meter that continuously displays the value of any analog channel. Except for choosing a Cell Assistant channel to bind to, all of the properties for this control affect the appearance of the control. Once a channel is selected, this control can also take advantage

of any Cell Assistant limits that have been created for this channel. Using limits is explained in a later section.

DigitalMeter

The Digital Meter is a simple numerical readout of any analog channel. Except for choosing a Cell Assistant channel to bind to, all of the properties for this control affect the appearance of the control. Once a channel is selected, this control can also take advantage of any Cell Assistant limits that have been created for this channel. Using limits is explained in a later section.

BarMeter

The Bar Meter is another popular meter style that is used to continuously display the value of any analog channel. Except for choosing a Cell Assistant channel to bind to, all of the properties for this control affect the appearance of the control. Once a channel is selected, this control can also take advantage of any Cell Assistant limits that have been created for this channel. Using limits is explained in a later section.

TextMeter

The Text Meter is used to display messages. This meter can be configured to run in any 1 of 3 different modes of operation. First of all, it can be connected to any Cell Assistant text channel such that the value of the text channel is displayed. A running TestPlan can change the displayed text by writing new text data to the appropriate text channel. The second mode of operation is to connect this meter to a digital channel. In this case, the meter will always display 1 of 2 text strings depending on the state of the digital channel. Finally, the third mode of operation is to connect this meter to a limit. In this case, the meter will always display 1 of 2 text string depending of the trip state of the limit.

PilotLight

The PilotLight control changes color depending on the state of the connected channel. This control has two modes of operation. First, it can be connected to a digital channel; it will change color based on the state of this digital channel. Secondly, it can be connected to a limit and will change color based on the trip state of the selected limit.

PushButton

The PushButton control is both an input device and an output device. It is connected to a digital output channel in Cell Assistant such that pressing this button will drive the state of this channel. Keep in mind that this button will drive the state of this channel just as a running TestPlan in Cell Assistant can drive the state of this channel programmatically using DSBasic commands. Thus, if this digital output channel is a digital output channel for a hardware device, pressing this button may cause commands to be sent to an external device.

The PushButton control can be configured to be a “maintained” or “momentary” pushbutton. A maintained pushbutton will toggle the state of the connected digital output channel whenever the pushbutton is pressed and released. A momentary button will set the connected digital output channel to the TRUE state only while the pushbutton is held down.

The PushButton control also has a digital input feedback channel associated with it. The value of this channel determines the color and text of the button. Using a separate feedback channel in this way allows the creation of closed-loop button operations. If the feedback feature is not needed, the digital output channel can also be used as the feedback channel.

StripChart

The StripChart control allows up to 4 Analog channels to be continuously plotted in real time. The Y-axis scale information for only one channel is displayed at a time. Click the ***Channel*** button to cycle through the Y-axis information for the other channels. The color of this information will match the color of the appropriate channel on the stripchart.

This control performs lots of calculations in order to get all data to be optimally drawn in the overall bounding rectangle of the control. At times, the width of the stripchart can change drastically (e.g. be reduced by 50 percent) for no obvious reason. For example, the channel name is changed slightly and now the width of the stripchart is halved. This can be confusing at times; so here is an explanation of what is happening.

Given the size of the bounding rectangle, the control first computes all margins (top, bottom, left, and right) that is used for X and Y axis information, captions, etc … This is determined by the length of these text strings and the selected fonts and font sizes. Then the required margins are subtracted from the overall bounding rectangle of the control, which results in a smaller rectangle that is used to draw the actual stripchart. The control software then looks at the selected time duration of the stripchart and determines the widest chart that it can optimally draw in this rectangle while still maintaining integer pixel per data point resolutions. This explains why something as simple as adding 1 letter to a channel name can reduce the width of the stripchart by 50 percent. Perhaps the stripchart was currently being drawn a 2 pixels per data point and now that the channel name has widened by 1 letter, the left margin must increase accordingly. Now there is only enough room to draw the stripchart at a 1.9 pixel per data point resolution which is not allowed. So the new resolution is set to 1 pixel per data point. Reducing the time span of the X-axis a bit can often restore the width of the stripchart. Adjusting the overall width of the entire StripChart control can also restore the width of the stripchart.

Also keep in mind that the left margin is determined by looking at the longest and tallest strings of all enabled channels since enough display real estate must be allocated to accommodate the worst case of all the channels.

ComputerSelect

The ComputerSelect control is used to create a VIP that runs over a network. The absence/presence of this control determines whether this particular VIP tries to access data on the local computer or a remote computer on the network. In the absence of a ComputerSelect control, all VIP Controls attempt to connect to a Cell Assistant data channel that exists on the local computer. As soon as a ComputerSelect control is added to a Visual Basic form, all controls rely on the ComputerSelect control for their data.

This control has two properties: The ***Computer*** name and the ***TCP/IP Port*** of the remote computer that this control should try to connect to. You can enter either the name of the computer or the IP address of the computer. If the remote computer exists on a local Intranet, you can also click the ***Browse*** button to get a list of available computers and then select the desired computer graphically.

The ***TCP/IP Port*** must match the TCP/IP port that is currently be used by the CA Data Server service that is running on the remote computer of interest. Typically, the default value should be specified.

If Cell Assistant is running on the specified computer (and it is configured to support network connections), this control will immediately connect to the remote computer. When the ComputerSelect control is first placed on a form, it will read “Not Connected”. Once a connection is established, the controls appearance will change to a progress bar that will provide visual feedback of network activity when the VIP is run.

At this point, you can continue to develop and/or or run VIP’s in the usual manner. Keep in mind that you are now connected to a remote Cell Assistant computer via a network connection that may have long network latencies. These delays are usually quite tolerable if running entirely in a local Intranet; however, these delays can be longer if the remote computer is farther away (in a network sense). For example, both the local and remote computers may be linked to the Internet via telephone connections to a local Internet Service Provider (ISP).

Selecting a Channel

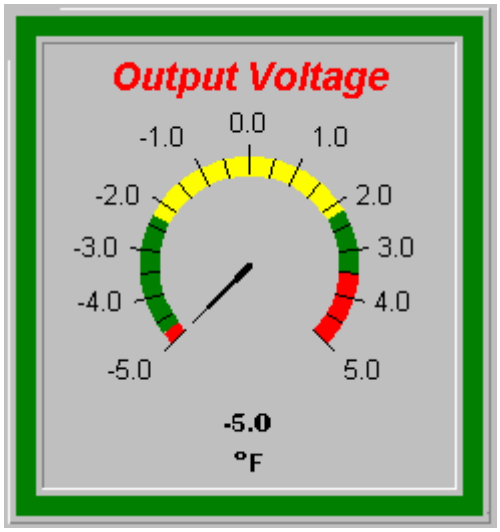
All of the VIPCONTROLS are designed to work exclusively with Cell Assistant. Each of the controls has one or more listboxes that display a list of channels that the control can be connected to. When constructing VIP’s that use these controls, always make sure that data acquisition is running in Cell Assistant. This causes a list of all available channels to be exposed to other applications (including these controls).

Using Channel Properties

By default, many of the appearance properties of each control are initially determined by the channel that they are initially connected to. For example, captions, units, numerical precision of readouts and scale numbering, On/Off text, etc … initially are taken from the Cell Assistant defaults for the connected channel. For each property, however, you can uncheck the “use channel default” box and enter your own value.

Using Limits

Many of the meters can also make use of Cell Assistant limit information. The limits that are available depend on the selected channel. Only limits that are based on the selected channel can be used. The limits affect how the meter scales are drawn. Meter scales are always the color green by default. Up to four available limits can be selected. Once selected, each limit must be assigned a color of red or yellow. The control will then repaint part of the scale based on the assigned color and limit values of each selected limit. An example is shown below.



In this example, three of the limits that were based on the selected channel were also selected to be used for this control also. One is a high limit with a setpoint of 3.5. The color red was assigned. Another was a low limit with a setpoint of –4.75. Again, the color red was assigned. Finally, an "in-range" limit with setpoints of –2.25 and 2.25 was selected and assigned the color yellow.

To this point, the only advantage of selecting limits is to provide a user with visual feedback about limit information for a channel by painting limit ranges on the meter scale. The real power of using limits with a meter is explained in the next section where it is explained how the meter border color will change when one of the selected limits trips in Cell Assistant.

Meter Borders

The border color of a meter is always colored green just like the meter scale; however, if limits are selected, the border color can be used as a highly visual indicator of limit trip conditions. If one of the selected limits trips, the border will change to the same color that has been assigned to that particular limit. The border will stay that color until the limit untrips.

Note that the border color has NOTHING to do with where the needle is pointing. For example, the needle could be pointing in the red range, yet Cell Assistant might not have that particular limit currently enabled. Thus, the limit will not trip and the border will not change color. Similarly, a limit can trip in Cell Assistant which causes the border color of all affected meters to change color. The actual value of the channel may then change which causes the needle to now point to a green portion of the scale. But the Cell Assistant limit handler (in DSBasic code) may still be processing the limit and has not yet allowed the limit trip state to return to normal. Thus, the border colors remain in their tripped state.

Chapter 11 - Utilities

The Cell Assistant software package also includes several other applications that can be used to view data. These utilities are typically launched from Cell Assistant (from the ***View*** menu) as previously described in this manual. They can also be launched using other common methods (i.e. from the Windows NT Explorer, creating desktop shortcuts, etc …).

Channel Viewer Utility

The Channel Viewer Utility (CHView.exe) is installed in the main Cell Assistant directory. It is typically launched from the Cell Assistant ***View*** menu (as explained in a previous chapter). Data Acquisition and/or a TestPlan must be started prior to launching the Channel Viewer.

This application presents a continuously updated listbox of all defined channels in the currently running TestSystem. It is mainly useful for troubleshooting and debugging. The display is continuously updated as long as data acquisition is running in Cell Assistant. The contents of the listbox can be filtered using the same series of checkboxes as described for the ***Channels*** page. Reducing the number of displayed channels is useful when the TestSystem contains many channels. Multiple instances of the Channel Viewer can also be launched; each can have different filter settings as needed.

Perhaps one of the most powerful features of the Channel Viewer is its ability to modify channel values. Double-click on any channel item in the listbox and the display filter area will be replaced with channel edit controls. Enter a new value for the channel and press the ***Accept*** button to update the channel data or the ***Cancel*** button to restore the display filter controls.

There are several issues to be aware of when editing channel data using this application. First of all, if you edit any input channel (Analog Input, Digital Input, etc …), the new value will only persist until the Cell Assistant data acquisition subsystem resamples that channel. For example, if you edit a channel value that is updated at 10 Hz, your data will only persist for 100 mSec (or less). This may not be long enough for the Channel Viewer display to update and display this new value. As you can see, it really makes no sense to edit “input” channels.

Another issue deals with editing “output” channels. This is a more typical use of the channel edit feature of this application. Always be aware of the fact that each output channel is mostly likely linked to a hardware device. Writing data to the channel will cause the appropriate data to be sent to the hardware device, so make sure you always understand what you are writing to a channel. Writing to a channel in the Channel Viewer has the same effects as programmatically writing to a channel in a DSBasic TestPlan. Note that data acquisition must be running in Cell Assistant in order for channel editing to be effective.

CSV File Viewer Utility

The CSV File Viewer Utility (CsvFileView.exe) is installed in the main Cell Assistant directory. It is typically launched from the Cell Assistant ***View*** menu (as explained in a previous chapter) in order to view the contents of a data file or flight recorder file.

This application is essentially a very simple spreadsheet-like application that has been optimized for Cell Assistant data files. The column widths are optimized. It is capable of reading larger data files

and it can load these larger data files at faster rate that most standard spreadsheet programs. On the other hand, this application does not have the many hundreds of other formatting and editing features of a spreadsheet program such as MicroSoft Excel. This application merely provides additional column resizing ability and simple zoom and print capabilities. More sophisticated editing of the data files should be done with Microsoft Excel (or a similar product).

Limit Set Viewer Utility

The Limit Set Viewer Utility (LimitSetView.exe) is installed in the main Cell Assistant directory. It is typically launched from the Cell Assistant ***View*** menu (as explained in a previous chapter). Data Acquisition and/or a TestPlan must be started prior to launching the Limit Set Viewer.

This application displays all limit setpoints for all limit sets. The format of the data is identical to the format of the Limit Set configuration page (as described in a previous chapter). The data can only be viewed by this application; it cannot be edited. The grid shows the current values of all limit sets. The column that represents the currently active limit set is shown in red. The contents of the grid are updated as needed if any of the limit setpoint values are updated by a running TestPlan.

Chapter 12 - Troubleshooting

This chapter provides solutions to some of the more common problems that a user of Cell Assistant may encounter. If you cannot find a solution to your problem here, please don't hesitate to call Dyne Systems at the number mentioned in Chapter 1.

Device Communications Error Dialog Boxes

Device communications error dialog boxes appear whenever there is a communications problem with any device. **Read the error dialog box carefully since it contains valuable information**. It should also be noted at this point that all communications errors are also logged in the system event log. This creates a permanent record of the error that can be viewed later on if needed.

Many errors are "timeout" errors. This occurs when a command is sent to a device and no response is received after a reasonable amount of time. This is usually caused by improper COM port settings and/or cabling problems. Always verify the physical port connection and all port settings first.

Some devices can report errors to Cell Assistant. In this case, the serial port connection is working properly; it is the device that is reporting a device-detected error. The error dialog box clearly differentiates "connection" related errors from errors that are returned from the device. The response from most devices is in ASCII format and is displayed in the error dialog box. At this point, you need to be familiar with the device that reported the error. You will also need documentation that explains the various error codes that each device can return. In most instances, the TestPlan needs to be modified because it is sending a commands to a device in an invalid order.

Some devices simply have idiosyncrasies that are difficult to work around in the DSBasic TestPlan. The hardware device drivers supplied by DyneSystemsCo. make every attempt to handle these problems and shield the TestPlan from them. Please don't hesitate to call DyneSystemsCo. if you are having a bizarre problem with any device. We can help you understand your device and assist in developing a work around.

Failures When Loading a TestPlan

Every version of Cell Assistant is able to load TestSystems that were created with previous versions of Cell Assistant. The reverse is not true. Occasionally a user of an older version of Cell Assistant will be given a TestPlan that was created with a new version. This results in an "Invalid file format" error message with the TestPlan is loaded. You should contact DyneSystemsCo. at this point in order to get an upgrade to Cell Assistant.

In other instances, a TestSystem will load but one or more devices may fail to initialize. This is usually caused by invalid serial port settings (e.g. the COM port is not defined or is already used by another device).

Watchdog Trips Occur

Customers that purchased the hardware watchdog option may occasionally get spurious trips of the watchdog hardware circuit. The root cause of these trips is due to the fact that the Windows NT operating system is NOT a true "hard real-time" operating system. The Cell Assistant software makes

every effort to update the external watchdog in a timely fashion to keep it from expiring (i.e. tripping) but in some situations, this is not possible. Here are some remedies for this problem.

1. Don't run other unnecessary applications while Cell Assistant is running a TestPlan. Some applications use excessive amounts of CPU time which prevents Cell Assistant from keep the watchdog hardware updated.
2. Certain DSBasic commands (i.e. OLE Automation, network file I/O, etc …) can tie up the CPU for long periods of time which prevents the remainder of Cell Assistant from updating the watchdog hardware. A special DSBasic command (SuspendWatchdog) has been created that allows a TestPlan to suspend the watchdog software module for short amounts of time. The SuspendWatchdog command (see Appendix B) should be called prior to any CPU consuming commands that are repeatedly causing the watchdog to trip.
3. The problem may simply be an underpowered computer and/or insufficient memory.
4. Call DyneSystemCo. for help.

CSV File Viewer Cannot Load Entire Data File

The CSV File Viewer application can read a maximum of 131,072 rows (128K) and 512 columns which is adequate for most applications (Microsoft Excel limits are 65,536 rows (64K) and 256 columns).

The column limit of 512 should always be sufficient since the widest row in any data file is equal to the number of channels that are logged (plus a few extra columns for time, date, etc …). Even the largest TestSystems written so far still have less than 200 channels (and hence data file columns).

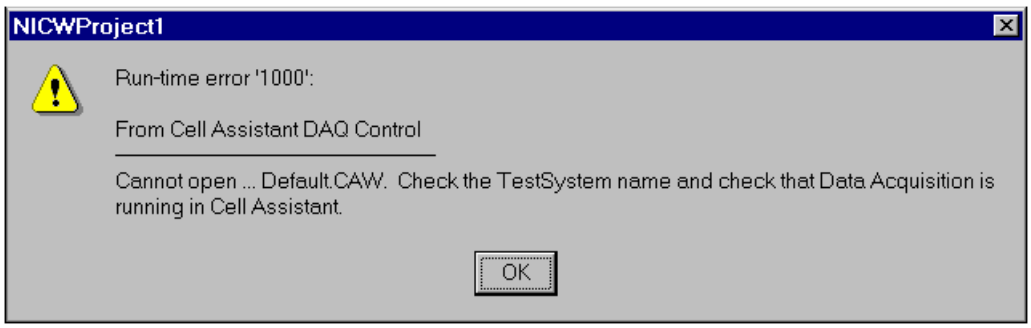
The row limit can easily be exceeded for high sampling rates for long periods of time. Some common sense must be applied when choosing the sampling rate. If logging at 20 Hz, the row limit will be reached in approximately 100 minutes (depending on the amount of other information that is logged to the file in addition to the data samples). Reducing the sampling rate to 10 Hz will result in a data file that hits the row limit in about 3.5 hours.

Anyway, use the following equation to roughly calculate how long you can log data at a specific rate while keeping the row count of the data file small enough to allow the file to be read in its entirety.

Duration (seconds) = 131,072 / sample rate (in Hz)

Getting VIP Errors From the CADAQ Control

Virtual Instrument Panels that use the CADAQ OLE custom control will terminate and display an error dialog box if the CADAQ control detects an error. A typical CADAQ error dialog box is shown below. Most errors are caused by programming mistakes that can easily be corrected as the VIP is being developed. Some errors can still occur later on even though the VIP had apparently been thoroughly debugged and had been running for a long time already. All possible CADAQ error codes are discussed below along with possible causes and solutions.


NICWProject1
Run-time error '1000':
From Cell Assistant DAQ Control
Cannot open ... Default.CAW. Check the TestSystem name and check that Data Acquisition is running in Cell Assistant.
OK

- Run-time error 1000: This error occurs when the OpenTestSystem( ) method is executed in Visual Basic and the CADAQ control cannot find a currently running TestSystem with the specified TestSystem name. If this error occurs during development, check the spelling of the TestSystem name. If this error occurs later on, check that Cell Assistant currently has a TestSystem with the specified name loaded and running (i.e. Data Acquisition has been started). The most frequent cause of this error is actually a problem with the settings in the NT Explorer application. Make sure that the "hide file extensions" property is unchecked in the NT Explorer (or File Manager). If this box is accidentally checked, the CADAQ control is looking for a full TestSystem name (e.g. Default.CAW) yet Cell Assistant will appear to be running a TestSystem without the extension (e.g. Default without the .CAW)
- Run-time error 1001: This error occurs when the RegisterChannel( ) method is executed in Visual Basic. This most likely cause of this error is a spelling mistake in the specified channel name. If the error occurs at a later date, the specified channel make have been deleted from the TestSystem in which case the VIP needs to be updated.
- Run-time error 1002: This error occurs when trying to access channel data with a handle that was NOT obtained from the RegisterChannel( ) method. This is a programming problem that must be fixed.
- Run-time error 1003: This error occurs when trying to access channel data using an incompatible method. For example, trying to write text data to a digital channel make no sense and will generate this error. Other invalid combinations are also possible.
- Run-time error 1004: This error typically occurs during development when the VIP is still running but Cell Assistant is constantly being stopped and restarted. When a VIP registers a channel, the CADAQ control determines exactly where (in computer memory) to find the specified channel data. If data acquisition in Cell Assistant is stopped and restarted, the channel data for the specified channel may no longer be in the same place (in computer memory). Anyway, the CADAQ control detects when this happens and transparently tries to re-register all of the currently registered channels. This re-registration process will fail if channels have been renamed and/or deleted. Again, this error only occurs during development if you develop a habit of starting and stopping data acquisition in Cell Assistant while VIP's are still running.

Problems Associated with NT User Administration

Cell Assistant saves some program settings on a "per user" basis. For example, the "most recently used" file list, and the size and location of the Cell Assistant main window are unique for each user. Thus, if you log into NT has a different user, don't be alarmed if the main Cell Assistant window is

now in a different position and the all of the TestSystems that you frequently load are no longer on the ***File*** drop-down menu.

A bigger problem associated with running Cell Assistant under different NT user accounts comes from NT itself. Windows NT also saves most settings on a "per user" basis. Some settings – if set incorrectly – can adversely affect the operation of Cell Assistant. For example, the NT Explorer has a "hide file extensions" option. Cell Assistant (an other applications as well) require this option to be unchecked (i.e. do NOT hide file extensions). This is explained the Appendix on proper computer setup for Cell Assistant. Anyway, if you have created a new NT user and are going to run Cell Assistant when logged in as this new user, please read over the Appendix on Computer setup and verify that all settings for this new user account are set correctly.

New Device is Detected Whenever Computer is Restarted

When the Windows NT operating system starts up, it automatically checks each serial port to see if a serial mouse is connected. If the DynLoc is incorrectly set up, it may respond to this port test which results in the NT operating system thinking there is a mouse connected to that port. In this case, NT will install a mouse driver (if available) or display a dialog box that asks the operator to provide the NT CD-ROM in order to get the mouse driver.

The solution is to disconnect the serial cable between the computer and the Dyn-Loc and restart your computer. Once NT is completely started, reconnect the serial cable. Launch Cell Assistant and load a TestSystem that includes the Dyn-Loc/DTC hardware device and go to the ***Diagnose Dyn-Loc*** configuration page. Enter "EQ" in the edit box and click the ***Send*** button. The DynLoc will most likely reply with an "EQ 0" response. Enter "EQ 1" in the edit box and click the ***Send*** button again. This permanently configures the Dyn-Loc such that is will not respond to the NT mouse detection routine at startup time.

Communcation Errors from the Dyn-Loc Diagnostic Page

This problem is merely another symptom of the previously described problem. When Windows NT boots up, it trys to detect hardware devices on each of the serial ports. The serial data that NT sends can (and usually does) generate errors in the Dyn-Loc. These errors remain queued up in the Dyn-Loc until later on when Cell Assistant begins to send valid commands the Dyn-Loc, at which time they are reported to Cell Assistant. This is not a problem during normal operation of Cell Assistant since all queued errors are automatically cleared every time Data Acquisition and/or a TestPlan is started.

This problem occurs (after booting NT) if a TestSystem is loaded and the operator immediately goes to the ***Hardware Devices*** page and activates the ***Diagnose Dyn-Loc*** page. In this case, all stored Dyn-Loc errors are NOT cleared out; hence, the next 20 (or so) commands that are sent to the Dyn-Loc from this page will result in a communications error. The solution is the enter "CQ" in the edit box and click the ***Send*** button. This will clear out all errors in the Dyn-Loc.

Sharing Hardware Devices with Other Computers

Specifically, many customers share a Pierburg Emissions bench between multiple Cell Assistant computers that run simultaneously. The serial port connection is frequently disconnected and

connected to another computer. This switching can cause spurious data bytes to be generated which can get sent to the Cell Assistant computer and/or the emissions bench. Special data channels have been created for the Pierburg Emissions Bench driver in order initiate the cleaning up and testing of the COM port connection after each switch. See Appendix A for a description of these channels. Pay special attention to the note that says that this test should NEVER be performed while the device is being sampled or unpredictable results (i.e. program crashes) may occur. This note easily overlooked. It is a simple matter to avoid this issue by making sure that emissions bench sampling is turned off first.

Dr. Watson

The Dr. Watson utility is a default debugger that ships with every copy of Windows NT. It is typically launched when a program exits abnormally (e.g. memory access violations). At the present time, the following situations have been identified with the Cell Assistant software that cause abnormal termination of the program (and hence the displaying of the Dr. Watson debugger dialog).

- ASCII File Driver: An access violation may occur if the ASCII file driver is being used and Cell Assistant is terminated while a network timeout is in progress. The following sequence will typically generate an access violation. Start data acquisition (or run a TestPlan) that causes the ASCII file driver to begin accessing a file that resides on another computer via the network. Disconnect the network cable from the local computer and then quickly exist the TestPlan. Cell Assistant will appear to be “frozen” while any pending network timeouts expire (could be as long as 40 seconds). Click on the exit icon in the upper right-hand corner of the Cell Assistant main window. As soon as the network timeout expires, an access violation may occur. This access violation is harmless. No loss of data will occur.

Appendix A - Device Channels

Allen Bradley SLC 5/03 PLC Channels

No	Name	Type	Command	Note	Comments
01	B10:0/0	DO		5	
02	B10:1/0	DO		5	
03	B10:2/0	DO		5	
04	B10:3/0	DO		5	
05	Clear Driver Error	DO		2	
01	B9:0/0	DI		1	
02	B9:0/1	DI		1	
03	B9:0/2	DI		1	
04	B9:0/3	DI		1	
05	B9:0/4	DI		1	
06	B9:0/5	DI		1	
07	B9:0/6	DI		1	
08	B9:0/7	DI		1	
09	B9:0/8	DI		1	
10	B9:0/9	DI		1	
11	B9:0/10	DI		1	
12	B9:0/11	DI		1	
13	B9:0/12	DI		1	
14	B9:0/13	DI		1	
15	B9:0/14	DI		1	
16	B9:0/15	DI		1	
01	B10:4	AO			
02	B10:5	AO			
03	B10:6	AO			
04	B10:7	AO			
01	B9:1	AI		1	
02	B9:2	AI		1	
03	B9:3	AI		1	
04	B9:4	AI		1	
01	Driver Error Type	DSI		2,3	0=No errors, 1=Send Failure, 2=Receive timeout, 3=Error returned from device
02	Additional Error Code	DSI		4	Valid if Driver Error Type > 0. See Note.
01	Driver Error Text	TI		2	Valid if Driver Error Type > 0

NOTES:

1) All digital and analog input values are continuously sampled at the selected sampling rate.
2) These channels are used by a TestPlan to check for various types of device communications errors. Typically, a limit is created and enabled that checks if the "Device Error Type" input becomes nonzero. If so, the TestPlan can handle the error based on the actual type. The "Driver Error Text" can be read and logged to a data file if desired. The driver latches the first error that occurs. Write any value to the "Clear Driver Error" channel to clear the current error and allow the next error that occurs to be latched.
3) An error dialog box is displayed whenever the "Driver Error Type" becomes nonzero. This essentially halts the driver (i.e. no more commands are sent to the device) until the operator responds by pressing the "Retry" or "Ignore" button. If "Ignore" is pressed and the command that failed was a sampling request (i.e. not initiated from the TestPlan), sampling for this device will be disabled for the remainder of the TestPlan.
4) This channel provides additional information whenever the "Driver Error Type" becomes nonzero.

If "Driver Error Type" is set to 1 or 2, then ...

0 = No additional error information is available.
1 = Timed out waiting for data.
2 = Timed out waiting for terminator character.
3 = Insufficient characters were received.
4 = Serial Port not defined.

If "Driver Error Type" is set to 3, then ...
0 = No additional error information is available.
N = STS code returned from the PLC (see the DF1 Protocol Manual for a list of possible STS codes).

5) Cell Assistant Digital Output channels are mapped to the least significant bit a 16-bit PLC data file word; however, when Cell Assistant set or resets a digital output bit, it actually sets or resets all bits in the respective 16-bit word. Thus, the PLC programmer should not attempt to use the upper 15 bits for any other purposes in the PLC program.

GENERAL NOTES:
The exact channel names are dependent on the device settings. The channels names listed above are generated when the device settings are as follows.

Type	Value
No. of Digital Inputs	16
No. of Analog Inputs	4
No. of Digital Outputs	4
No. of Analog Outputs	4

All channel names reference word/bit locations in the PLC. Cell Assistant can only read data in Data File # 9 in the PLC and write data into Data File #10 in the PLC. These files are user-defined files that must be created by the PLC programmer (file type = Binary). The size of each file must be large enough to accommodate all data channels. Digital Input channels are mapped to individual bits in the PLC and, thus, each 16-bit word in Data File # 9 contains 16 digital input bits that can be read by Cell Assistant. Digital Output channels are mapped to the least significant bit of each 16-bit word in Data File #10. This is somewhat inefficient but is necessary because the SLC 5/03 CPU does not support any commands that allows Cell Assistant to modify an individual bit in a 16-bit word. Analog Input and Analog Outputs are also mapped to 16-bit words in the respective data files. In this example, the data files must be configured to contain the following number of elements:

Data File #9 = 16/16 + 4 = 5 words.
Data File #10 = 4 + 4 = 8 words

As you can see from the created channel names, channel data is stored in the following order in the PLC data files:

1) Digital Inputs Stored first in Data File #9.
2) Analog Inputs Stored immediately after Digital Inputs in Data File #9.
3) Digital Outputs Stored first in Data File #10.
4) Analog Outputs Stored immediately after Digital Outputs in Data File #10.

The resulting channel names match the naming conventions used by RSLogix 500 ™, the standard programming software package used to program SLC 500 series PLC's. Thus, the name of each channel clearly indicates which word/bit it is mapped to in the PLC.

NOTE: It is extremely important to make initial allocations that are larger than the initial design would normally dictate. In fact, the initiall allocations should anticipate all future needs. Since all channel names are created based on these allocations, it is important that the allocation never be changed. If they are, a severe amount of PLC and CA4W TestPlan reprogramming may be required.

VERSION INFORMATION:
DyneSystemsCo., LLC Document Section Version 0.02 January 25, 1999

ASCII File Channels

No	Name	Type	Command	Note	Comments
01	Clear Driver Error	DO		3	
01	File Not Accessible	DI		6	If = TRUE, file is currently NOT being sampled.
02	File Is Static	DI		7	If = TRUE, file is NOT being updated.
01	Analog Input 001	AI		1,2	
02	Analog Input 002	AI		1,2	
03	Analog Input 003	AI		1,2	
04	Analog Input 004	AI		1,2	
05	Analog Input 005	AI		1,2	
06	Analog Input 006	AI		1,2	
07	Analog Input 007	AI		1,2	
08	Analog Input 008	AI		1,2	
09	Analog Input 009	AI		1,2	
10	Analog Input 010	AI		1,2	
11	Seconds Since Last File Access	AI		6	Valid if "File Not Accessible" channel = TRUE. See Note. Otherwise, always reads as 0.
12	Seconds Since Static File Detected	AI		7	Valid if "File Is Static" channel = TRUE. See Note. Otherwise, always reads as 0.
01	Driver Error Type	DSI		3,4	0=No errors, 1=File access error
02	Additional Error Code	DSI		5	Valid if Driver Error Type > 0. See Note.
01	Driver Error Text	TI		3	Valid if Driver Error Type > 0

NOTES:

1) The previous table lists 10 Analog Input channels. The actual number of channels that are available is configurable in the driver by changing the **No. of channels in file** setting. See General Notes (below) for more information.
2) These input channel values are continuously sampled at the rate that is specified in the driver settings. The sampling rates for this driver are typically slow compared to other device drivers; therefore, this driver requires you to specify the "sampling interval" (in seconds) instead of the sampling rate (in Hz). The driver allows you to select a sampling interval of 1, 2, 5, 10, 20, or 30 seconds or 1, 2, 3, 4, 5, or 10 minutes. NOTE: For all other drivers, Cell Assistant acquires the data for each channel at the same rate that the driver acquires the data from the device. This is not true for this driver. Cell Assistant is designed to sample data from the drivers at 1 Hz or greater; at this time, it cannot sample data from the drivers at rates of 0.5 Hz, 0.2 Hz, 0.1 Hz, 0.05 Hz, etc … (which correspond to the specified allowable sampling intervals). Instead, the drivers report a sampling rate of 1 Hz to Cell Assistant for all channels regardless of the actual sampling rate. Thus, Cell Assistant will sample all ASCII driver channels at 1 Hz even though the driver may be acquiring the data at a lower rate.
3) These channels are used by a TestPlan to check for various types of file access errors. Typically, a limit is created and enabled that checks if the "Device Error Type" input becomes nonzero. If so, the TestPlan can handle the error based on the actual type. The "Driver Error Text" can be read and logged to a data file if desired. The driver latches the first error that occurs. Write any value to the "Clear Driver Error" channel to clear the current error and allow the next error that occurs to be latched.
4) An error dialog box is displayed whenever the "Driver Error Type" is set to 1. This essentially halts the driver (i.e. no more file accesses are attempted) until the operator responds by pressing the "Retry" or "Ignore" button. If "Ignore" is pressed, sampling for this device will be disabled for the remainder of the TestPlan.
5) This channel provides additional information whenever the "Driver Error Type" becomes nonzero.

   If "Driver Error Type" is set to 1, then ...
   0 = No additional error information is available.
   1 = Failed to open file.

6) When file access errors occur, an error dialog box is displayed and the watchdog hardware (if any) will be tripped; however, in some installations, file access errors are expected to occur during normal operation. The driver contains a check box on the Settings page that (if checked) tells the driver to ignore file access errors. In this case, an error dialog box will NOT be displayed and the watchdog hardware will continue to be updated. In this mode of operation, the "File Not Accessible" and "Seconds Since Last File Access" channels should be used. When a file access error occurs, the "File Not Accessible" channel is set = TRUE. This can be used to inform a running TestPlan that the Analog Input values from the data file are no longer being updated. The "Seconds Since Last File Access" channel can then be used to monitor (and possibly display to an operator) how long it has been since the Analog Input channel values were last updated.
7) These channels are used to detect if the external application that is supposed to be periodically writing data to this file has stopped updating the file for some reason. Every time the file data is sampled, the time and date of the file is compared to the time and date of the file from the previous sample. If the time and date are the same, the "File Is Static" channel is set = TRUE. The "Seconds Since Static File Detected" channel can then be used to monitor how long it has been since the file was last updated.

GENERAL NOTES:
The number of Analog Input channels that are available to a TestPlan is determined by the device driver settings. The above table lists 10 Analog Input channels because the **No. of channels in file** setting has been set to 10. This setting can be set to any value between 0 and 200 (inclusive). If the specified ASCII file contains more channel data than specified in the driver setting, the additional data will be ignored. If the ASCII file contains less channel data than specified in the driver setting, the driver will set the additional channels to value of 0.0.

Even though this driver is capable of accessing files that reside on other computers via a network connection, it was NOT designed with that purpose in mind. In other words, it is designed to access files that exist on the local computer that is running Cell Assistant. Accessing files over a network can potentially introduce network delays and network timeouts that will compromise the real-time effieciency of the Cell Assistant software.

VERSION INFORMATION:
DyneSystemsCo., LLC Document Section Version 0.03 May 15, 1998

CHANNEL TYPE ABBREVIATIONS:

DO	= Digital Output	DI	= Digital Input
AO	= Analog Output	AI	= Analog Input
TO	= Text Output	TI	= Text Input
DCO	= Device Control Output	DSI	= Device Status Input

AVL Fuel Balance Channels

No	Name	Type	Command	Note	Comments
01	Set Filling Valve State	DO	SAVA SAVZ		Set = TRUE to open valve. Set = FALSE to close valve.
02	Set Bypass Valve State	DO	SBVA SBVZ		Set = TRUE to open valve. Set = FALSE to close valve.
03	Remote	DO	SREM		
04	Manual	DO	SMAN		
05	Standby	DO	STBY		
06	Reset	DO	SRES		
07	Initialize Measurement Mode (SMES)	DO	SMES		
08	Start Measuring	DO	SINT		
09	Stop Measuring	DO	SMIS		
10	Get Interval Measurement (AIMW)	DO	AIMW		
11	Get Total Interval Measurement (AIMG)	DO	AIMG		
12	Get Result (AERG)	DO	AERG		
13	Clear Driver Error	DO		7	
14	Clear AK Data Warning	DO		10	
01	Remote/Master	DI		1	If = TRUE, in remote mode. If = FALSE, in manual mode.
02	Filling Valve State	DI		1	If = TRUE, valve is open. If = FALSE, valve is closed.
03	Bypass Valve State	DI		1	If = TRUE, valve is open. If = FALSE, valve is closed.
01	Mean Measuring Value (AIMW) kg/h	AI		2	
02	Averaging Time (AIMW) sec	AI		2	
03	Fuel Consumption (AIMW) g	AI		2	
04	Mean Measuring Value (AIMG) kg/h	AI		3	
05	Averaging Time (AIMG) sec	AI		3	
06	Fuel Consumption (AIMG) g	AI		3	
07	Current Filling Level (AWRT) g	AI		4	
08	Instantaneous Value (AWRT) kg/h	AI		4	
09	Fuel Consumption (AWRT) g	AI		4	
10	Averaging Time (AWRT) sec	AI		4	
11	Measuring Value (AERG) kg/h	AI		5	
12	Measuring Time (AERG) sec	AI		5	
13	Fuel Consumption (AERG) g	AI		5	
01	Current Device Mode	DSI		1	0=SRES, 1=STBY, 2=SMES, 3=SINT, 4=SMZV, 5=SMGV, 6=SATK, 7=SKCH
02	Driver Error Type	DSI		7,8	0=No errors, 1=Send Failure, 2=Receive timeout, 3=Error returned from device
03	Additional Error Code	DSI		9	Valid if Driver Error Type > 0. See Note.
04	AK Data Warning	DSI		10	0=No warnings, 1='#' character(s) detected in data
01	Send String (To Device)	TO		6	
01	Response String (From Device)	TI		6	
02	Driver Error Text	TI		7	Valid if Driver Error Type > 0
03	AK Data Warning Text	TI		10	Valid if AK Data Warning > 0

NOTES:

1) These input channel values are continuously sampled at a rate of 1 Hz by sending the ASTZ command to this device. At the present time, the sampling rate cannot be changed by the user.
2) These input values are updated by sending the AIMW command (i.e. write any value to the "Get Interval Measurement" channel).
3) These input values are updated by sending the AIMG command (i.e. write any value to the "Get Total Interval Measurement" channel).
4) These input channel values are continuously sampled at a rate of 1 Hz by sending the AWRT command to this device. At the present time, the sampling rate cannot be changed by the user.
5) These input values are updated by sending the AERG command (i.e. write any value to the "Get Result" channel).
6) These commands are used to send any string to this device and retrieve the response. Some of the AK protocol details are automatically handled for you. For example, 2 bytes are appended to the start of your string (STX and a "space") and one byte is added to the end or your string (ETX) before it is send to this device. Similarly, these 3 bytes (2 at the beginning and 1 at the end) are stripped from the response string that is returned via the "Response String" channel.

7) These channels are used by a TestPlan to check for various types of device communications errors. Typically, a limit is created and enabled that checks if the “Device Error Type” input becomes nonzero. If so, the TestPlan can handle the error based on the actual type. The “Driver Error Text” can be read and logged to a data file if desired. The driver latches the first error that occurs. Write any value to the “Clear Driver Error” channel to clear the current error and allow the next error that occurs to be latched.
8) An error dialog box is displayed whenever the “Driver Error Type” is set to 1 or 2. This essentially halts the driver (i.e. no more commands are sent to the device) until the operator responds by pressing the “Retry” or “Ignore” button. If “Ignore” is pressed and the command that failed was a sampling request (i.e. not initiated from the TestPlan), sampling for this device will be disabled for the remainder of the TestPlan. An error dialog box is NOT displayed if the “Driver Error Type” is set to 3. The TestPlan must check for this condition in a limit handler and process accordingly.
9) This channel provides additional information whenever the “Driver Error Type” becomes nonzero.

If “Driver Error Type” is set to 1 or 2, then ...

0 = No additional error information is available.
1 = Timed out waiting for data.
2 = Timed out waiting for terminator character.
3 = Insufficient characters were received.
4 = Serial Port not defined.

If “Driver Error Type” is set to 3, then ...

0 = No additional error information is available.
1 = AK response contains “????” (Unknown or erroneous command)
2 = AK response contains “NA” (Selected function not available)
3 = AK response contains “BS” (System is busy)
4 = AK response contains “OF” (System not in computer mode)
5 = AK response contains “DF” (Data error)
6 = AK response contains “SE” (Syntax error)
7 = AK response contains a Nonzero Error Status. Limit handler should send an ASTF command to query error codes.
8 = AK response is corrupted and/or invalid (e.g. <STX> character is missing)

10) These channels are used by a TestPlan to monitor for ‘#’ characters in responses from this device because response data values can sometimes be preceded and/or replaced by ‘#’ characters under various conditions. Typically, a limit is created and enabled that checks if the “AK Data Warning” input becomes nonzero. If so, the TestPlan can handle the warning as needed. The “AK Data Warning Text” can be read and logged to a data file if desired. The driver latches the first warning that occurs. Write any value to the “Clear AK Data Warning” channel to clear the current warning and allow the next warning that occurs to be latched.

GENERAL NOTES:
The previous notes mention two different commands that are sent to this device at a 1 Hz rate for each command. These alternating commands are sent at 500 mSec intervals such that the overall sampling rate of this device is 2 Hz; yet, the effective sampling rate of each command remains 1 Hz.

VERSION INFORMATION:
DyneSystemsCo., LLC Document Section Version 0.11 September 5, 1999

AVL 439 Opacimeter Channels

No	Name	Type	Command	Note	Comments
01	Reset	DO	SRES K0		
02	Control Mode	DO	SREM K0		
03	Function Off	DO	SIDL K0		
04	Pause	DO	SPAU K0		
05	Zeroing	DO	STBY K0		
06	Start Measurement	DO	SMGA K0		
07	Clear Driver Error	DO		5	
01	Device NOT Ready	DI		8	TRUE = Device is NOT ready to start measurements.
01	Measurement Value	AI		1,2,3	See note 3.
02	Gas Temperature (°C)	AI		1,2	
03	Measuring Chamber Pressure (mbar)	AI		1,2	
04	Gas Flow Rate (liters/min)	AI		1,2	
01	Driver Error Type	DSI		5,6	0=No errors, 1=Send Failure, 2=Receive timeout, 3=Error returned from device
02	Additional Error Code	DSI		7	Valid if Driver Error Type > 0. See Note.
01	Send String (To Device)	TO		4	
01	Response String (From Device)	TI		4	
02	Driver Error Text	TI		5	Valid if Driver Error Type > 0

NOTES:

1) These input channel values are continuously sampled at a rate of 1Hz. The driver sends the AKON and AMDT commands to the Opacimeter to retrieve these values.
2) If a particular data value is not available when a query is made, the Opacimeter will return a '#' character instead of the expected data. When this occurs, the driver will report a data value of 0.0 to Cell Assistant. Similarly, if a data value is only conditionally valid, the Opacimeter will prefix the data value with a '#' character. In this case, the driver will report the returned data value to Cell Assistant.
3) The "Measurement Value" value must be configured using the EPAR command (see AVL 439 Operating Manual). The EPAR command can be sent to the Opacimeter from the Diagnostic dialog page in Cell Assistant or from a running TestPlan using the "Send String (To Device)" channel.
4) These commands are used to send any string to this device and retrieve the response. Some of the AK protocol details are automatically handled for you. For example, 2 bytes are appended to the start of your string (STX and a "space") and one byte is added to the end or your string (ETX) before it is sent to this device. Similarly, these 3 bytes (2 at the beginning and 1 at the end) are stripped from the response string that is returned via the "Response String (From Device)" channel.
5) These channels are used by a TestPlan to check for various types of device communications errors. Typically, a limit is created and enabled that checks if the "Device Error Type" input becomes nonzero. If so, the TestPlan can handle the error based on the actual type. The "Driver Error Text" can be read and logged to a data file if desired. The driver latches the first error that occurs. Write any value to the "Clear Driver Error" channel to clear the current error and allow the next error that occurs to be latched.
6) An error dialog box is displayed whenever the "Driver Error Type" becomes nonzero. This essentially halts the driver (i.e. no more commands are sent to the device) until the operator responds by pressing the "Retry" or "Ignore" button. If "Ignore" is pressed and the command that failed was a sampling request (i.e. not initiated from the TestPlan), sampling for this device will be disabled for the remainder of the TestPlan.
7) This channel provides additional information whenever the "Driver Error Type" becomes nonzero.

   If "Driver Error Type" is set to 1 or 2, then ...
   0 = No additional error information is available.
   2 = Timed out waiting for terminator character.
   3 = Insufficient characters were received.
   4 = Serial Port not defined.

   If "Driver Error Type" is set to 3, then ...
   Always = 0. No additional error information is available.

8) When the Opacimeter is zeroing, it will return error code 14 (Not ready) in response to all commands. The device driver filters out these errors (i.e. no communications error dialog box will be displayed) and, instead, presents the Ready/Not Ready status to Cell Assistant using this digital input channel.

GENERAL NOTES:
None.

VERSION INFORMATION:
DyneSystemsCo., LLC Document Section Version 0.02 August 20, 1999

Baldor Series 22H Vector Drive Channels

No	Name	Type	Command	Note	Comments
01	Stop	DO	S		Writing any value sends command.
02	Enable Drive	DO	E		Writing any value sends command.
03	Disable Drive	DO	D		Writing any value sends command.
04	Clear Fault Code	DO	C	5	Writing any value sends command.
05	Get Fault Code and Text	DO	F	5	Writing any value sends command.
06	Clear Driver Error	DO		6	
01	Opto Output #1	DI		1	Output from drive, Input to Cell Assistant Parameter 1501, pin 19
02	Opto Output #2	DI		1	Output from drive, Input to Cell Assistant Parameter 1502, pin 20
03	Opto Output #3	DI		1	Output from drive, Input to Cell Assistant Parameter 1503, pin 21
04	Opto Output #4 (FAULT)	DI		1,3,5	Output from drive, Input to Cell Assistant Parameter 1504, pin 22
01	RPM Setpoint	AO	Vnnnn		Puts drive in Velocity mode. Commands velocity setpoint to nnnn
02	Torque Setpoint	AO	Tnnnn		Puts drive in Torque mode. Commands torque setpoint to nnnn
03	Accel Time #1	AO	P1101 nnnn.n		Valid range of nnnn.n is 0.0 to 3600.0
04	Decel Time #1	AO	P1102 nnnn.n		Valid range of nnnn.n is 0.0 to 3600.0
01	Motor RPM	AI		2	
02	Motor Frequency (Hz)	AI		2	
03	Motor Voltage	AI		2	
04	Motor Current (Amps)	AI		2	
05	Analog Out #1	AI		1	Output from drive, Input to Cell Assistant Parameter 1508, pin 6
06	Analog Out #2	AI		1	Output from drive, Input to Cell Assistant Parameter 1509, pin 7
01	Fault Code	DSI		5	
02	Driver Error Type	DSI		6,7	0=No errors, 1=Send Failure, 2=Receive timeout, 3=Error returned from device
03	Additional Error Code	DSI		8	Valid if Driver Error Type > 0. See Note.
01	Send String with Reply (To Device)	TO		4	
02	Send String no Reply (To Device)	TO		4	
01	Response String (From Device)	TI		4	
02	Fault Text	TI		5	
03	Driver Error Text	TI		6	Valid if Driver Error Type > 0

NOTES:

1) These input channel values are continuously sampled at a rate of 1 Hz by sending the “IO” command to this device. At the present time, the sampling rate cannot be changed by the user.
2) These input channel values are continuously sampled at a rate of 1 Hz by sending the “O” command to this device. At the present time, the sampling rate cannot be changed by the user.
3) The drive opto output #4 is used for fault indication. Be sure that the drive is programmed for OPTO OUTPUT #4 to be “FAULT” (parameter 1504).
4) These channels are used to send any command string to this device. The TestPlan writer must know which Baldor commands return replies and which do not. The appropriate “Send” command should then be used. If you accidentally send a command that does NOT return a reply using the “Send String with Reply (To Device)” channel, a communications timeout error will occur. The reply to any command can be read back via the “Response String (From Device)” channel.
5) These channels are used to retrieve fault code information from this device. Typically, a limit is created that monitors the “Opto Output #4 (FAULT)” channel. When tripped, the “Get Fault Code and Text” channel is used to retrieve fault information from this device. Fault information can then be retrieved using the “Fault Code” and “Fault Text” channels.
6) These channels are used by a TestPlan to check for various types of device or communications errors. Typically, a limit is created and enabled that checks if the “Device Error Type” input becomes nonzero. If so, the TestPlan can handle the error based on the actual type. The “Driver Error Text” can be read and logged to a data file if desired. The driver latches the first error that occurs. Write any value to the “Clear Driver Error” channel to clear the current error and allow the next error that occurs to be latched.

7) An error dialog box is displayed whenever the “Driver Error Type” becomes nonzero. This essentially halts the driver (i.e. no more commands are sent to the device) until the operator responds by pressing the “Retry” or “Ignore” button. If “Ignore” is pressed and the command that failed was a sampling request (i.e. not initiated from the TestPlan), sampling for this device will be disabled for the remainder of the TestPlan.
8) This channel provides additional information whenever the “Driver Error Type” becomes nonzero.

If “Driver Error Type” is set to 1 or 2, then ...
0 = No additional error information is available.
1 = Timed out waiting for data.
2 = Timed out waiting for terminator character.
3 = Insufficient characters were received.
4 = Serial Port not defined.
7 = Timed out waiting for a line of data.
8 = Insufficient lines of data were received.

If “Driver Error Type” is set to 3, then ...
0 = No additional error information is available.
1 = Errors occurred while scanning response for the expected data.

GENERAL NOTES:
None.

VERSION INFORMATION:
DyneSystemsCo., LLC Document Section Version 0.04 July 13, 2000

CAI Emissions Bench Channels

No	Name	Type	Command	Note	Comments
01	Standby	DO	STBY		
02	Flow Zero Gas	DO	SNGA		
03	Flow Sample Gas	DO	SMGA		
04	Flow Background Sample Gas	DO	SBAK		
05	Flow Span Gas	DO	SEGA		
06	Purge	DO	SSPL		
07	Zero Span Cycle	DO	SZSC		
08	Get Device Status	DO	ASTZ	2	
09	Sample Enable	DO		6	
10	Clear Driver Error	DO		3	
01	Select Sample Line	DCO	SLI<n>		n = 1 or 2
01	Concentration – HHC	AI		1	
02	Concentration – COM	AI		1	
03	Concentration – NOX	AI		1	
04	Concentration – CO2	AI		1	
05	Concentration – COH	AI		1	
06	Concentration – O2	AI		1	
01	Bench Mode	DSI		2	1=STBY, 2=SNGA, 3=SMGA, 4=SBAK, 5=SEGA, 6=SSPL, or 7=SZSC.
02	Driver Error Type	DSI		3,4	0=No errors, 1=Send Failure, 2=Receive timeout, 3=Error returned from device
03	Additional Error Code	DSI		5	Valid if Driver Error Type > 0. See Note.
01	Send Command (To Device)	TO		7	
01	Get Response (From Device)	TI		7	
02	Driver Error Text	TI		3	Valid if Driver Error Type > 0

NOTES:

1) These input channel values are sampled at a rate of 0.5 Hz (i.e. every 2 seconds) using the AKON command. Note that Cell Assistant will display a sampling rate of 1 Hz even though the driver is only acquiring the data for these channels at a rate of 0.5 Hz.
2) These input channel values are not continuously sampled. They can be updated by sending the ASTZ command to the device (i.e. write any value to the "Get Device Status" channel).
3) These channels are used by a TestPlan to check for various types of device communications errors. Typically, a limit is created and enabled that checks if the "Device Error Type" input becomes nonzero. If so, the TestPlan can handle the error based on the actual type. The "Driver Error Text" can be read and logged to a data file if desired. The driver latches the first error that occurs. Write any value to the "Clear Driver Error" channel to clear the current error and allow the next error that occurs to be latched.
4) An error dialog box is displayed whenever the "Driver Error Type" becomes nonzero. This essentially halts the driver (i.e. no more commands are sent to the device) until the operator responds by pressing the "Retry" or "Ignore" button. If "Ignore" is pressed and the command that failed was a sampling request (i.e. not initiated from the TestPlan), sampling for this device will be disabled for the remainder of the TestPlan.
5) This channel provides additional information whenever the "Driver Error Type" becomes nonzero.

   If "Driver Error Type" is set to 1 or 2, then ...
   0 = No additional error information is available.
   1 = Timed out waiting for data.
   2 = Timed out waiting for terminator character.
   3 = Insufficient characters were received.
   4 = Serial Port not defined.

   If "Driver Error Type" is set to 3, then ...
   Always = 0 (No additional error information is available at this time).

6) This channel is used to enable/disable data acquisition (i.e. sampling) for this device. This allows a running TestPlan to control when and how often this device is sampled.
7) These commands are used to send any command string to this device and retrieve the response. Some of the AK protocol details are automatically handled for you. For example, 2 bytes are appended to the start of your string (STX and a "space") and one byte is added to the end or your string (ETX) before it is sent to this device. Similarly, these 3 bytes (2 at

the beginning and 1 at the end) are stripped from the response string that is returned via the “Get Response (From Device)” channel.

GENERAL NOTES:
None.

VERSION INFORMATION:
DyneSystemsCo., LLC Document Section Version 0.02 August 26, 1999

Companion Channels

No	Name	Type	Command	Note	Comments
01	Clear Driver Error	DO		2	
01	Analog Output :: Module 01	AO			
02	Analog Output :: Module 02	AO			
03	Analog Output :: Module 03	AO			
04	Analog Output :: Module 04	AO			
05	Analog Output :: Module 05	AO			
06	Analog Output :: Module 06	AO			
07	Analog Output :: Module 07	AO			
08	Analog Output :: Module 08	AO			
01	Backplane 1 :: Module 01	AI		1	
02	Backplane 1 :: Module 02	AI		1	
03	Backplane 1 :: Module 03	AI		1	
04	Backplane 1 :: Module 04	AI		1	
05	Backplane 1 :: Module 05	AI		1	
06	Backplane 1 :: Module 06	AI		1	
07	Backplane 1 :: Module 07	AI		1	
08	Backplane 1 :: Module 08	AI		1	
09	Backplane 1 :: Module 09	AI		1	
10	Backplane 1 :: Module 10	AI		1	
11	Backplane 1 :: Module 11	AI		1	
12	Backplane 1 :: Module 12	AI		1	
13	Backplane 1 :: Module 13	AI		1	
14	Backplane 1 :: Module 14	AI		1	
15	Backplane 1 :: Module 15	AI		1	
16	Backplane 1 :: Module 16	AI		1	
01	Driver Error Type	DSI		2,3	0=No errors, 1=Send Failure, 2=Receive timeout, 3=Error returned from device
02	Additional Error Code	DSI		4	Valid if Driver Error Type > 0. See Note.
01	Driver Error Text	TI		2	Valid if Driver Error Type > 0

NOTES:

1) These input channel values are continuously sampled at the rate specified in the driver settings. The maximum sampling rate is 20 Hz. True anti-aliased input signal data is only guaranteed if the sampling rate is set to 20 Hz. Aliasing can occur at lower sampling rates (which may be tolerable in some applications).
2) These channels are used by a TestPlan to check for various types of device communications errors. Typically, a limit is created and enabled that checks if the “Device Error Type” input becomes nonzero. If so, the TestPlan can handle the error based on the actual type. The “Driver Error Text” can be read and logged to a data file if desired. The driver latches the first error that occurs. Write any value to the “Clear Driver Error” channel to clear the current error and allow the next error that occurs to be latched.
3) An error dialog box is displayed whenever the “Driver Error Type” is set to 1 or 2. This essentially halts the driver (i.e. no more commands are sent to the device) until the operator responds by pressing the “Retry” or “Ignore” button. If “Ignore” is pressed and the command that failed was a sampling request (i.e. not initiated from the TestPlan), sampling for this device will be disabled for the remainder of the TestPlan.
4) This channel provides additional information whenever the “Driver Error Type” becomes nonzero.

If “Driver Error Type” is set to 1 or 2, then ...

0 = No additional error information is available.
1 = Timed out waiting for data.
2 = Timed out waiting for terminator character.
3 = Insufficient characters were received.
4 = Serial Port not defined.

If “Driver Error Type” is set to 3, then ...

0 = No additional error information is available.
1 = DAQ board failed to initialize.
2 = INFO file is missing - Call Dyne Systems.
3 = INFO file is corrupted - Call Dyne Systems.

GENERAL NOTES:

The above table lists only 8 analog output channels; however, the Companion system actually supports 16 analog output channels. Analog output modules are installed in a multiplexed backplane and connected to the system as backplane 5. Similarly, the above table lists only 16 analog input channels; yet, the Companion system supports 96 analog input channels. Analog input modules are installed in non-multiplexed backplanes (up to 16 modules per backplane) and are connected to the system as backplanes 1, 2, 3, 4, 6, or 7 (backplane 5 is used for analog output modules).

VERSION INFORMATION:
DyneSystemsCo., LLC Document Section Version 0.01 August 1, 1997

Daytronic System 10 Channels

No	Name	Type	Command	Note	Comments
01	Digital1	DO			Subchannel of 10AIO-16 Digital Card.
02	Digital2	DO			" " " " "
03	Clear Driver Error	DO		1	
01	Digital1	DI			Subchannel of 10AIO-16 Digital Card.
02	Digital2	DI			" " " " "
01	VoltageInput1	AI			Subchannel of 10A64-8 Voltage Card.
02	VoltageInput2	AI			" " " " "
03	Thermocouple1	AI			Subchannel of 10A10-4 Thermocouple Card.
04	Thermocouple2	AI			" " " " "
05	StrainGage1	AI			Subchannel of 10A70-2 Strain Gage Card.
06	StrainGage2	AI			" " " " "
07	FrequencyInput1	AI			Subchannel of 10A41-2 Frequency Input Card.
08	FrequencyInput2	AI			" " " " "
01	Driver Error Type	DSI		1,2	0=No errors, 1=Send Failure, 2=Receive timeout, 3=Error returned from device
02	Additional Error Code	DSI		3	Valid if Driver Error Type > 0. See Note.
01	Driver Error Text	TI		1	Valid if Driver Error Type > 0

NOTES:

1) These channels are used by a TestPlan to check for various types of device communications errors. Typically, a limit is created and enabled that checks if the "Device Error Type" input becomes nonzero. If so, the TestPlan can handle the error based on the actual type. The "Driver Error Text" can be read and logged to a data file if desired. The driver latches the first error that occurs. Write any value to the "Clear Driver Error" channel to clear the current error and allow the next error that occurs to be latched.
2) An error dialog box is displayed whenever the "Driver Error Type" becomes nonzero. This essentially halts the driver (i.e. no more commands are sent to the device) until the operator responds by pressing the "Retry" or "Ignore" button. If "Ignore" is pressed and the command that failed was a sampling request (i.e. not initiated from the TestPlan), sampling for this device will be disabled for the remainder of the TestPlan.
3) This channel provides additional information whenever the "Driver Error Type" becomes nonzero.

If "Driver Error Type" is set to 1 or 2, then ...

0 = No additional error information is available.
1 = Timed out waiting for data.
2 = Timed out waiting for terminator character.
3 = Insufficient characters were received.
4 = Serial Port not defined.

If "Driver Error Type" is set to 3, then ...

0 = No additional error information is available.

GENERAL NOTES:

All digital input (DI) and analog input (AI) channels are sampled at the selected sampling rate.

The list shown above only gives 2 examples of each subchannel for the specified cards which is enough to demonstrate the default channel names and numbering conventions (e.g. the 10A64-8 actually has 8 voltage input subchannels but only 2 are listed above).

The channel names of each subchannel are really only the "default" names that are assigned by the device driver when the card is added to the system. The name of each subchannel can be uniquely assigned in the driver setup dialog box. You must exercise some amount of caution if you decide to alter the channel names at some later date. Once a subchannel name is referenced in a TestSystem (i.e. a virtual channel has been created that references this subchannel name), the name should no longer be changed. If you do decide to rename, remove, and/or add subchannels, you MUST review all of your TestSystems to make sure all referenced channels still exist (and are what you expect them to be).

VERSION INFORMATION:

DyneSystemsCo., LLC Document Section Version 0.09 July 16, 1997

Direct Logic 205 PLC Channels

No	Name	Type	Command	Note	Comments
01	Digital Output : C0	DO			
02	Digital Output : C1	DO			
03	Digital Output : C2	DO			
04	Digital Output : C3	DO			
05	Digital Output : C4	DO			
06	Digital Output : C5	DO			
07	Digital Output : C6	DO			
08	Digital Output : C7	DO			
09	Clear Driver Error	DO		2	
01	Digital Input : C10	DI		1	
02	Digital Input : C11	DI		1	
03	Digital Input : C12	DI		1	
04	Digital Input : C13	DI		1	
05	Digital Input : C14	DI		1	
06	Digital Input : C15	DI		1	
07	Digital Input : C16	DI		1	
08	Digital Input : C17	DI		1	
01	Analog Output : V1410	AO			
02	Analog Output : V1411	AO			
03	Analog Output : V1412	AO			
04	Analog Output : V1413	AO			
01	Analog Input : V1400	AI		1	
02	Analog Input : V1401	AI		1	
03	Analog Input : V1402	AI		1	
04	Analog Input : V1403	AI		1	
01	Driver Error Type	DSI		2,3	0=No errors, 1=Send Failure, 2=Receive timeout, 3=Error returned from device
02	Additional Error Code	DSI		4	Valid if Driver Error Type > 0. See Note.
01	Driver Error Text	TI		2	Valid if Driver Error Type > 0

NOTES:

1) All digital and analog input values are continuously sampled at the selected sampling rate.
2) These channels are used by a TestPlan to check for various types of device communications errors. Typically, a limit is created and enabled that checks if the "Device Error Type" input becomes nonzero. If so, the TestPlan can handle the error based on the actual type. The "Driver Error Text" can be read and logged to a data file if desired. The driver latches the first error that occurs. Write any value to the "Clear Driver Error" channel to clear the current error and allow the next error that occurs to be latched.
3) An error dialog box is displayed whenever the "Driver Error Type" becomes nonzero. This essentially halts the driver (i.e. no more commands are sent to the device) until the operator responds by pressing the "Retry" or "Ignore" button. If "Ignore" is pressed and the command that failed was a sampling request (i.e. not initiated from the TestPlan), sampling for this device will be disabled for the remainder of the TestPlan.
4) This channel provides additional information whenever the "Driver Error Type" becomes nonzero.

If "Driver Error Type" is set to 1 or 2, then ...

0 = No additional error information is available.
1 = Timed out waiting for data.
2 = Timed out waiting for terminator character.
3 = Insufficient characters were received.
4 = Serial Port not defined.

If "Driver Error Type" is set to 3, then ...

0 = No additional error information is available.
N = Exception code returned from the PLC (see PLC manual for a list of valid exception codes).

GENERAL NOTES:

The exact channel names are dependent on the device settings. The channels names listed above are generated when the device settings are as follows.

Type	First Relay [octal]	Start Address [octal]	No. of Outputs	No. of Inputs
Control Relays	0		8	8
Analog Inputs		1400	4	
Analog Outputs		1410	4	

As you can see, digital output and digital input control relays are consecutively numbered. The relays are always contiguous where the digital input control relays always start immediately after the digital output control relays. Clearly, the **First Relay** device setting is essentially the starting control relay number of the first digital output. The starting control relay number of the first digital input is equal to **First Relay + No. of Outputs**. Note that control relay numbering is in octal (octal numbering is used in all PLC documentation and programming software from Direct Logic).

This is a unique device driver in that the “names” of the channels can be altered by changing the device settings. You must exercise some amount of caution if you decide to alter the device settings at some later date. All channels in all TestSystems should also be reviewed and updated at that time since some channels may now “disappear” or worse, the channel NAME still exists, but its FUNCTION is completely different.

The “command” column in the first table is deliberately left blank because all PLC communications are performed using binary communication packets which is not documented here. The Modbus protocol is used. If you do have a Modicon manual available (and are interested), digital input channel reads are performed using a “Read Coil Status” packet (opcode = 01). Digital output channel writes are performed using a “Force Single Coil” packet (opcode = 05). Analog output channel writes are performed using a “Preset Single Register” packet (opcode = 06). Analog input channel reads are performed using a “Read Holding Registers” packet (opcode = 03).

VERSION INFORMATION:
DyneSystemsCo., LLC Document Section Version 0.01 December 31, 1998

Drivers Trace Channels

No	Name	Type	Command	Note	Comments
01	Enable Graph	DO	E 1 E 0	1	Set = TRUE to enable graph and enable sampling. Set = FALSE to disable graph and disable sampling.
02	Reset Graph	DO	R		
03	Clear Driver Error	DO		2	
01	Reached End of Data	DI		1	If = TRUE, driver has reached last data point in file.
01	Set Grade	AO	G <grade>		Grade range is -90.0 to +90.0 degrees (inclusive).
01	Message Background Color	DCO	C <color>	6	0=Normal (Gray), 1=Red, 2=Green
01	Driver Error Type	DSI		2,3	0=No errors, 1=Send Failure, 2=Receive timeout, 3=Error returned from device
02	Additional Error Code	DSI		4	Valid if Driver Error Type > 0. See Note.
01	Load Data File	TO	L <filename>	5	
02	Set Message Line 1	TO	1 <message>		
03	Set Message Line 2	TO	2 <message>		
01	Driver Error Text	TI		2	Valid if Driver Error Type > 0

NOTES:

1) These input channel values are continuously sampled at a rate of 1 Hz by sending the "." character to this device. At the present time, the sampling rate cannot be changed by the user. Sampling begins when the "Enable Graph" channel is set to TRUE and stops when this channel is set to FALSE.
2) These channels are used by a TestPlan to check for various types of device communications errors. Typically, a limit is created and enabled that checks if the "Device Error Type" input becomes nonzero. If so, the TestPlan can handle the error based on the actual type. The "Driver Error Text" can be read and logged to a data file if desired. The driver latches the first error that occurs. Write any value to the "Clear Driver Error" channel to clear the current error and allow the next error that occurs to be latched.
3) An error dialog box is displayed whenever the "Driver Error Type" is set to 1, 2, or 3. This essentially halts the driver (i.e. no more commands are sent to the device) until the operator responds by pressing the "Retry" or "Ignore" button. If "Ignore" is pressed and the command that failed was a sampling request (i.e. not initiated from the TestPlan), sampling for this device will be disabled for the remainder of the TestPlan.
4) This channel provides additional information whenever the "Driver Error Type" becomes nonzero.

   If "Driver Error Type" is set to 1 or 2, then ...
   0 = No additional error information is available.
   1 = Timed out waiting for data.
   2 = Timed out waiting for terminator character.
   3 = Insufficient characters were received.
   4 = Serial Port not defined.

   If "Driver Error Type" is set to 3, then ...
   0 = No additional error information is available.

5) The specified filename can optionally contain full path information (e.g. D:\RoadLoad\Data\TestFile.txt) in which case the DriversTrace computer will attempt to open the file in the specified path. If only the filename is specified, the DriversTrace computer will look for the file in the default path which is stored in the system registry in the DriversTrace computer. This file contains roadload speed data at a resolution of 5 points per second (1 data point per line).
6) The background color of both message lines can be changed to indicate warning or alarm conditions.

GENERAL NOTES:
None.

VERSION INFORMATION:
DyneSystemsCo., LLC Document Section Version 0.02 April 30, 1998

DTC Channels

No	Name	Type	Command	Note	Comments
01	Control Power	DO	MD AUT,ON MD AUT,OFF		Set = TRUE to enable throttle control. Set = FALSE to disable throttle control.
02	Emergency Stop	DO	MD COMP,ESTOP		Writing any value sends ESTOP command.
03	Clear Driver Error	DO		3	
01	Reference	AO	MD AUTO, ON SP <value> 0x1f <low byte><high byte>		Set Reference = <value> (Binary setpoint version)
02	LAC	AO	LAC = <value> 0x1e <low byte><high byte>		Set LAC = <value> (Binary setpoint version)
01	Control Mode	DCO	MD AUT,POS MD AUT,RPM MD AUT,TQ MD AUT,VAC		Set = 0 to select POSITION mode. Set = 1 to select SPEED mode. Set = 2 to select TORQUE mode. Set = 3 to select VACUUM mode.
02	Reference Source	DCO	MD MAN MD AUT		Set = 0 to select MANUAL mode. Set = 1 to select AUTO mode.
01	DTC is ON	DI		1	
02	In POSITION Mode	DI		1	
03	In RPM Mode	DI		1	
04	In TORQUE Mode	DI		1	
05	In SETUP Mode	DI		1	
06	In AUTO Mode	DI		1	
07	In MANUAL Mode	DI		1	
08	E-STOP from Dyn-Loc	DI		1	
09	Overspeed/Underspeed from Dyn-Loc	DI		1	
10	Soft Overspeed Activated	DI		1	
11	Soft Shutdown Activated	DI		1	
12	Throttle Jog Function Activated	DI		1	
13	Waiting for CANCEL Key	DI		1	
14	Loss of Dyn-Loc Detected	DI		1	
15	Stall Detected	DI		1	
16	Stall Cycle Completed	DI		1	
01	DTC Power	AI		1	
02	DTC Speed	AI		1	
03	DTC Torque	AI		1	
04	DTC Throttle Position	AI		1	
01	Driver Error Type	DSI		3,4	0=No errors, 1=Send Failure, 2=Receive timeout, 3=Error returned from device
02	Additional Error Code	DSI		5	Valid if Driver Error Type > 0. See Note.
01	Send String (To Device)	TO		2	
01	Response String (From Device)	TI		2	
02	Driver Error Text	TI		3	Valid if Driver Error Type > 0

NOTES:

1) These input channel values are continuously sampled at the selected sampling rate for this device. The device driver sends an ENQ (0x05) character to the DTC in order to perform binary data acquisition.
2) These commands are used to send any string to this device and retrieve the response. The driver automatically appends a RET character (Carriage Return) to the end of your string before sending it to the device.
3) These channels are used by a TestPlan to check for various types of device communications errors. Typically, a limit is created and enabled that checks if the "Device Error Type" input becomes nonzero. If so, the TestPlan can handle the error based on the actual type. The "Driver Error Text" can be read and logged to a data file if desired. The driver latches the first error that occurs. Write any value to the "Clear Driver Error" channel to clear the current error and allow the next error that occurs to be latched.

4) An error dialog box is displayed whenever the “Driver Error Type” becomes nonzero. This essentially halts the driver (i.e. no more commands are sent to the device) until the operator responds by pressing the “Retry” or “Ignore” button. If “Ignore” is pressed and the command that failed was a sampling request (i.e. not initiated from the TestPlan), sampling for this device will be disabled for the remainder of the TestPlan.
5) This channel provides additional information whenever the “Driver Error Type” becomes nonzero.

If “Driver Error Type” is set to 1 or 2, then ...
0 = No additional error information is available.
1 = Timed out waiting for data.
2 = Timed out waiting for terminator character.
3 = Insufficient characters were received.
4 = Serial Port not defined.

If “Driver Error Type” is set to 3, then ...
0 = No additional error information is available.
N = A unique error code returned from the device. Check the DTC Users Manual.

GENERAL NOTES:
This driver is used when a Digital Throttle Controller (i.e. DTC) is directly connected to a COM port on the host computer. If the DTC is indirectly connected to the host PC via a Dyn-Loc, the Dyn-Loc/DTC device driver should be used to communicate with the DTC.

VERSION INFORMATION:
DyneSystemsCo., LLC Document Section Version 0.02 March 13, 1997

Dyn-Loc/DTC Channels

No	Name	Type	Command	Note	Comments
01	Control Power (Dyn-Loc)	DO	MD COMP,DON MD COMP,DOFF		Set = TRUE to enable loading/motoring. Set = FALSE to disable loading/motoring.
02	Control Power (DTC)	DO	THR MD AUT,ON THR MD AUT,OFF		Set = TRUE to enable throttle control. Set = FALSE to disable throttle control.
03	Emergency Stop	DO	MD COMP,EMS		Writing any value sends ESTOP command.
04	AutoZero (Dyn-Loc)	DO	AZ		Writing any value sends AZ command.
05	ShuntCal (Dyn-Loc)	DO	SC		Writing any value sends SC command.
06	Clear Driver Error	DO		3	
01	Reference (Dyn-Loc)	AO	SP <value> 0x1d <low byte><high byte>		Set Reference = <value> (Binary setpoint version)
02	LAC (Dyn-Loc)	AO	LAC <value> 0x1c <byte>		Set LAC = <value> (Binary setpoint version)
03	AutoSpan (Dyn-Loc)	AO	AS <value>		Autospan using <value>
04	Reference (DTC)	AO	THR MD AUTO, ON SP <value> 0x1f <low byte><high byte>		Set Reference = <value> (Binary setpoint version)
05	LAC (DTC)	AO	THR LAC <value> 0x1e <low byte><high byte>		Set LAC = <value> (Binary setpoint version)
01	Control Mode (Dyn-Loc)	DCO	MD COMP,RPM MD COMP,TQ		Set = 0 to select SPEED mode. Set = 1 to select TORQUE mode.
02	Reference Source (Dyn-Loc)	DCO	MD MSTR MD REM MD COMP		Set = 0 to select MASTER mode. Set = 1 to select REMOTE mode. Set = 2 to select COMPUTER mode.
03	Control Mode (DTC)	DCO	THR MD AUT,POS THR MD AUT,RPM THR MD AUT,TQ THR MD AUT,VAC		Set = 0 to select POSITION mode. Set = 1 to select SPEED mode. Set = 2 to select TORQUE mode. Set = 3 to select VACUUM mode.
04	Reference Source (DTC)	DCO	THR MD MAN THR MD AUT		Set = 0 to select MANUAL mode. Set = 1 to select AUTO mode.
01	Dyno Torque Sign	DI		1	Dyn-Loc Status bits.
02	Shunt-Cal in Progress	DI		1,7	
03	Auto-Span or Auto-Zero in Progress	DI		1,7	
04	Dyn-Loc Status Bit3	DI		1	
05	Power Supply Ok	DI		1	
06	DC Dyne	DI		1	
07	DTC Present	DI		1	
08	Dyn-Loc Status Bit7	DI		1	
09	Dyne On	DI		1	
10	LAC Complete	DI		1	
11	Computer Mode	DI		1	
12	Absorb Only Mode	DI		1	
13	OS/US Tripped	DI		1	
14	Control Locked	DI		1	
15	Control Mode	DI		1	
16	Emergency Stop Status	DI		1	
17	DTC is ON	DI		1,6	DTC Status bits.
18	In POSITION Mode	DI		1,6	
19	In RPM Mode	DI		1,6	
20	In TORQUE Mode	DI		1,6	
21	In SETUP Mode	DI		1,6	
22	In AUTO Mode	DI		1,6	
23	In MANUAL Mode	DI		1,6	
24	E-STOP from Dyn-Loc	DI		1,6	
25	Overspeed/Underspeed from Dyn-Loc	DI		1,6	
26	Soft Overspeed Activated	DI		1,6	
27	Soft Shutdown Activated	DI		1,6	
28	Throttle Jog Function Activated	DI		1,6	

29	Waiting for CANCEL Key	DI	1,6	
30	Loss of Dyn-Loc Detected	DI	1,6	
31	Stall Detected	DI	1,6	
32	Stall Cycle Completed	DI	1,6	
01	Dyno Power	AI	1	
02	Dyno Speed	AI	1	
03	Dyno Torque	AI	1	
04	Throttle Position	AI	1	
01	Driver Error Type	DSI	3,4	0=No errors, 1=Send Failure, 2=Receive timeout, 3=Error returned from device
02	Additional Error Code	DSI	5	Valid if Driver Error Type > 0. See Note.
01	Send String (To Device)	TO	2	
01	Response String (From Device)	TI	2	
02	Driver Error Text	TI	3	Valid if Driver Error Type > 0

NOTES:

1) These input channel values are continuously sampled at the selected sampling rate for this device. The device driver sends a single character to the Dyn-Loc in order to perform binary data acquisition. The binary command that is sent depends on the selected sampling rate and whether or not the Dyn-Loc has a DS503 board installed.

Command	When used
0x05	DS503 board NOT installed.
0x09	DS503 board installed, sampling rates 1, 2, or 5 Hz.
0x0A	DS503 board installed, sampling rates 10 or 20 Hz.

2) These commands are used to send any string to this device and retrieve the response. The driver automatically appends a RET character (Carriage Return) to the end of your string before sending it to the device.
3) These channels are used by a TestPlan to check for various types of device communications errors. Typically, a limit is created and enabled that checks if the "Device Error Type" input becomes nonzero. If so, the TestPlan can handle the error based on the actual type. The "Driver Error Text" can be read and logged to a data file if desired. The driver latches the first error that occurs. Write any value to the "Clear Driver Error" channel to clear the current error and allow the next error that occurs to be latched.
4) An error dialog box is displayed whenever the "Driver Error Type" becomes nonzero. This essentially halts the driver (i.e. no more commands are sent to the device) until the operator responds by pressing the "Retry" or "Ignore" button. If "Ignore" is pressed and the command that failed was a sampling request (i.e. not initiated from the TestPlan), sampling for this device will be disabled for the remainder of the TestPlan.
5) This channel provides additional information whenever the "Driver Error Type" becomes nonzero.

   If "Driver Error Type" is set to 1 or 2, then ...
   0 = No additional error information is available.
   1 = Timed out waiting for data.
   2 = Timed out waiting for terminator character.
   3 = Insufficient characters were received.
   4 = Serial Port not defined.

   If "Driver Error Type" is set to 3, then ...
   0 = No additional error information is available.
   N = A unique error code returned from the device. Check the Dyn-Loc/DTC Users Manual.

6) These channels are only valid if the Dyn-Loc has a DS503 board installed. This board supports additional binary commands that allow the DTC status bits to be continuously sampled along with the Dyn-Loc status bits, speed, torque, power, and throttle position. If a DS503 board is NOT installed, they will all read as "0" when accessed by any TestPlan.
7) These channels are only valid if the Dyn-Loc has a DS503 board installed. The Auto-Zero (AZ), Auto-Span (AS), and Shunt-Cal (SC) commands all return an immediate response even though the actual processing of the command is not yet complete. These bits can be polled by the TestPlan to monitor the status of the respective command. If a DS503 board is NOT installed, they will all read as "0" when accessed by any TestPlan.

GENERAL NOTES:
None.

VERSION INFORMATION:
DyneSystemsCo., LLC Document Section Version 0.13 June 18, 1997

Horiba EXSA-1500 Channels (GPIB Version)

No	Name	Type	Command	Note	Comments
01	Reset	DO	AOP,01,00		
02	Calibrate	DO	AOP,02,00		
03	Measure	DO	AOP,05,00		
04	Purge	DO	AOP,06,00		
05	Set NO Mode	DO	SCS,00		
06	Set NOX Mode	DO	SCS,01		
07	Get Data	DO	RMD	1	
08	Get Measuring Ranges	DO	RRS	2	
09	Get Purge Time	DO	RPT	3	
10	Get Calibration Times	DO	RCT	4	
11	Get Operation Status	DO	RFS	5	
12	Get NO/NOX Mode	DO	RCS	6	
13	Get Device Error Codes	DO	RET	7	
14	Get Device Alarm Codes	DO	RAT	7	
15	Sample Enable	DO		1	Set = TRUE to enable data acquisition (i.e. sampling) Set = FALSE to disable data acquisition.
16	Clear Device Errors/Alarms Exist	DO		7	
17	Clear Driver Error	DO		12	
01	Zero Gas	DCO	AOP,03,n	8	n = Channel number (range 00 to 05). (See Note)
02	Span Gas	DCO	AOP,04,n	8	n = Channel number (range 00 to 05). (See Note)
03	Get Span Gas Concentration	DCO	RSG,n	9	n = Channel number (range 01 to 05).
04	Set Range	DCO	SRS,n,m	11	n = Channel number (range 01 to 05). m = Range number (01 to 06 or 10 for Automatic ranging).
01	Device Errors and/or Alarms Exist	DI		7	If = TRUE, one or more Error and/or Alarm codes exist .
01	Channel 1	AI		1	
02	Channel 2	AI		1	
03	Channel 3	AI		1	
04	Channel 4	AI		1	
05	Channel 5	AI		1	
06	Sequence Number	AI		1	
07	AFR	AI		1	
08	LAMBDA	AI		1	
09	Purge Time	AI		3	
10	Zero(1) Time	AI		4	
11	Zero(2) Time	AI		4	
12	Span Time	AI		4	
13	Span Gas Concentration	AI		9	
01	Channel 1 Meas. Range	DSI		2	Measuring range value (01 to 06 or 11 for Automatic ranging)
02	Channel 2 Meas. Range	DSI		2	Measuring range value (01 to 06 or 12 for Automatic ranging)
03	Channel 3 Meas. Range	DSI		2	Measuring range value (01 to 06 or 13 for Automatic ranging)
04	Channel 4 Meas. Range	DSI		2	Measuring range value (01 to 06 or 14 for Automatic ranging)
05	Channel 5 Meas. Range	DSI		2	Measuring range value (01 to 06 or 15 for Automatic ranging)
06	Operation Status	DSI		5	01=Reset, 02=Calibration, 03=Zero, 04=Span, 05=Measure, 06=Purge
07	NO/NOX Mode	DSI		6	00=NO Mode, 01=NOX Mode
08	Driver Error Type	DSI		12,13	0=No errors, 1=Send Failure, 2=Receive Timeout, 3=Receive Failure
09	Additional Error Code	DSI		14	Valid if Driver Error Type > 0. See Note.
01	Send String (To Device)	TO		10	Use to send commands that do NOT have a response.
02	Send String (To Device) Get Response	TO		10	Use to send commands that provide a response.
01	Response String (From Device)	TI		10	
02	Device Error Codes	TI		7	
03	Device Alarm Codes	TI		7	
04	Driver Error Text	TI		12	Valid if Driver Error Type > 0

NOTES:

1) When sampling is enabled (i.e. the “Sample Enable” channel is set = TRUE), these input values are continuously sampled at a rate of 1 Hz by sending the “RMD” command to this device. If sampling is not enabled (i.e. the “Sample Enable” channel is set = FALSE), these input values can still be updated by writing any value to the “Get Data” channel which also sends the “RMD” command to this device.
2) These input values are updated by sending the “RRS” command (i.e. write any value to the “Get Measuring Ranges” channel).
3) The “Purge Time” input value is updated by writing any value to the “Get Purge Time” channel.
4) The “Zero(1) Time”, “Zero(2) Time”, and “Span Time” input values are updated by writing any value to the “Get Calibration Times” channel.
5) The “Operation Status” input value is updated by writing any value to the “Get Operation Status” channel.
6) The “NO/NOX Mode” input value is updated by writing any value to the “Get NO/NOX Mode” channel.
7) Whenever any error and/or alarm codes exist in this device, the “Device Errors and/or Alarms Exist” channel becomes TRUE. The TestPlan should then write a value to the “Get Device Error Codes” and “Get Device Alarm Codes” channels in order to retrieve all existing error and alarm codes. The returned error and alarm codes are then available via the “Device Error Codes” and “Device Alarm Codes” channels. The TestPlan is responsible for parsing and extracting the returned list of error codes from the “Device Error Codes” channel and the returned list of alarm codes from the “Device Alarm Codes” channel; however, the error and alarm code strings are typically just logged “as is” into a data file and are not parsed or processed in any way. When done, write any value to the “Clear Device Errors/Alarms Exist” channel such that the driver can now monitor for additional errors and/or alarms from this device.
8) The channel numbering is slightly different for these commands (specifically for channels 00 and 01). See the Horiba EXSA-1500 Instruction Manual for more information.
9) The “Span Gas Concentration” input value is updated by writing the proper value to the “Get Span Gas Concentration” channel.
10) These commands are used to send any string to this device and retrieve the response. The protocol details are automatically handled for you; i.e., the <CR><LF> characters are automatically appended to whatever string that is written to this channel. Similarly, these characters are stripped from the end of the response string that is returned via the “Response String” channel. Use the “Send String (To Device) and Get Response” channel if you are sending commands that provide a response; otherwise, use just the “Send String (To Device)” channel to send commands that do NOT have a response.
11) The data values written to this channel are in the form of “nm” (i.e. the ten’s digit is the channel number and the 1’s digit is the range number). For example, to set channel 2 to range 3, the TestPlan should write a 23 to the “SetRange” channel. If you wish to enable autoranging for a particular channel, the hundred’s digit is the channel number and the lower two digits are the range number. For example, to enable autoranging for channel 4, the TestPlan should write 410 to the “SetRange” channel.
12) These channels are used by a TestPlan to check for various types of device communications errors. Typically, a limit is created and enabled that checks if the “Device Error Type” input becomes nonzero. If so, the TestPlan can handle the error based on the actual type. The “Driver Error Text” can be read and logged to a data file if desired. The driver latches the first error that occurs. Write any value to the “Clear Driver Error” channel to clear the current error and allow the next error that occurs to be latched.
13) An error dialog box is displayed whenever the “Driver Error Type” becomes nonzero. This essentially halts the driver (i.e. no more commands are sent to the device) until the operator responds by pressing the “Retry” or “Ignore” button. If “Ignore” is pressed and the command that failed was a sampling request (i.e. not initiated from the TestPlan), sampling for this device will be disabled for the remainder of the TestPlan.
14) This channel provides additional information whenever the “Driver Error Type” becomes nonzero.

    0 or less (i.e. negative number) = Negated NI-GPIB error code returned from the controller card (see note below).
    2 = Timed out waiting for terminator character.
    5 = GPIB - device not initialized.
    6 = GPIB - timed out writing to device.

    NOTE: The National Instruments error codes range from 0 to 20 (at this time). In order to not have them conflict with typical Cell Assistant driver error codes, this driver negates them. Just keep this in mind if you attempt to look up a returned code using NI documentation.

GENERAL NOTES:
None.

VERSION INFORMATION:
DyneSystemsCo., LLC Document Section Version 0.02 December 9, 1997

Mini-Companion Channels

No	Name	Type	Command	Note	Comments
01	Clear Driver Error	DO		2	
01	Backplane 1 :: Module 01	AI		1	
02	Backplane 1 :: Module 02	AI		1	
03	Backplane 1 :: Module 03	AI		1	
04	Backplane 1 :: Module 04	AI		1	
05	Backplane 1 :: Module 05	AI		1	
06	Backplane 1 :: Module 06	AI		1	
07	Backplane 1 :: Module 07	AI		1	
08	Backplane 1 :: Module 08	AI		1	
09	Backplane 1 :: Module 09	AI		1	
10	Backplane 1 :: Module 10	AI		1	
11	Backplane 1 :: Module 11	AI		1	
12	Backplane 1 :: Module 12	AI		1	
13	Backplane 1 :: Module 13	AI		1	
14	Backplane 1 :: Module 14	AI		1	
15	Backplane 1 :: Module 15	AI		1	
16	Backplane 1 :: Module 16	AI		1	
01	Driver Error Code	DSI		2	0=No errors. 1=SCAN setup failed. 2=SCAN failed.
01	Driver Error Text	TI		2	Valid if Driver Error Code > 0

NOTES:

1) These input channel values are continuously sampled at 20 Hz by the device driver.
2) These channels are used by a TestPlan to check for various types of device errors. Typically, a limit is created and enabled that checks if the “Driver Error Code” input becomes nonzero. If so, the TestPlan can handle the error based on the actual type. The “Driver Error Text” channel can be read and logged to a data file if desired. The driver latches the first error that occurs. Write any value to the “Clear Driver Error” channel to clear the current error and allow the next error that occurs to be latched.
3) An error dialog box is displayed whenever the “Driver Error Code” becomes non-zero. This essentially halts the driver (i.e. sampling for this device will be disabled for the remainder of the TestPlan.

GENERAL NOTES:

None.

VERSION INFORMATION:

DyneSystemsCo., LLC Document Section Version 0.01 February 4, 1999

Modicon PLC Channels

No	Name	Type	Command	Note	Comments
01	Digital Output 1:00300	DO			
02	Digital Output 2:00301	DO			
03	Digital Output 3:00302	DO			
04	Digital Output 4:00303	DO			
05	Digital Output 5:00304	DO			
06	Digital Output 6:00305	DO			
07	Digital Output 7:00306	DO			
08	Digital Output 8:00307	DO			
09	Clear Driver Error	DO		2	
01	Digital Input 1:00308	DI		1	
02	Digital Input 2:00309	DI		1	
03	Digital Input 3:00310	DI		1	
04	Digital Input 4:00311	DI		1	
05	Digital Input 5:00312	DI		1	
06	Digital Input 6:00313	DI		1	
07	Digital Input 7:00314	DI		1	
08	Digital Input 8:00315	DI		1	
01	Analog Output 1:40100	AO			
02	Analog Output 2:40101	AO			
03	Analog Output 3:40102	AO			
04	Analog Output 4:40103	AO			
01	Analog Input 1:30200	AI		1	
02	Analog Input 1:30201	AI		1	
03	Analog Input 1:30202	AI		1	
04	Analog Input 1:30203	AI		1	
01	Driver Error Type	DSI		2,3	0=No errors, 1=Send Failure, 2=Receive timeout, 3=Error returned from device
02	Additional Error Code	DSI		4	Valid if Driver Error Type > 0. See Note.
01	Driver Error Text	TI		2	Valid if Driver Error Type > 0

NOTES:

1) All digital and analog input values are continuously sampled at the selected sampling rate.
2) These channels are used by a TestPlan to check for various types of device communications errors. Typically, a limit is created and enabled that checks if the "Device Error Type" input becomes nonzero. If so, the TestPlan can handle the error based on the actual type. The "Driver Error Text" can be read and logged to a data file if desired. The driver latches the first error that occurs. Write any value to the "Clear Driver Error" channel to clear the current error and allow the next error that occurs to be latched.
3) An error dialog box is displayed whenever the "Driver Error Type" becomes nonzero. This essentially halts the driver (i.e. no more commands are sent to the device) until the operator responds by pressing the "Retry" or "Ignore" button. If "Ignore" is pressed and the command that failed was a sampling request (i.e. not initiated from the TestPlan), sampling for this device will be disabled for the remainder of the TestPlan.
4) This channel provides additional information whenever the "Driver Error Type" becomes nonzero.

If "Driver Error Type" is set to 1 or 2, then ...

0 = No additional error information is available.
1 = Timed out waiting for data.
2 = Timed out waiting for terminator character.
3 = Insufficient characters were received.
4 = Serial Port not defined.

If "Driver Error Type" is set to 3, then ...

0 = No additional error information is available.

GENERAL NOTES:

The exact channel names are dependent on the device settings. The channels names listed above are generated when the device settings are as follows.

Type	Starting	No. of	No. of

	Address	Outputs	Inputs
Digital I/O Coils (0xxxx)	0300	8	8
Analog Input Registers (3xxxx)	0200	4	
Analog Output Registers (4xxxx)	0100	4	

As you can see, digital output and digital input coil addresses are consecutively numbered. The addresses are always contiguous where the digital input addresses always start immediately after the digital output addresses. Clearly, the **Starting Address** device setting is essentially the starting coil address of the first digital output coil. The starting coil address of the first digital input is equal to **Starting Address + No. of Outputs**.

This is a unique device driver in that the "names" of the channels can be altered by changing the device settings. You must exercise some amount of caution if you decide to alter the device settings at some later date. All channels in all TestSystems should also be reviewed and updated at that time since some channels may now "disappear" or worse, the channel NAME still exists, but its FUNCTION is completely different.

The "command" column in the first table is deliberately left blank because all Modicon PLC communications are performed using binary communication packets which is not documented here. If you do have a Modicon manual available (and are interested), digital input channel reads are performed using a "Read Coil Status" packet (opcode = 01). Digital output channel writes are performed using a "Force Single Coil" packet (opcode = 05). Analog output register writes are performed using a "Preset Single Register" packet (opcode = 06). Analog input register reads are performed using a "Read Input Registers" packet (opcode = 04).

VERSION INFORMATION:
DyneSystemsCo., LLC Document Section Version 0.11 January 28, 1998

Pierburg CVS Channels

No	Name	Type	Command	Note	Comments
01	Reset	DO	SRES		
02	Standby	DO	STBY		
03	Pause	DO	SPAU		
04	Manual	DO	SMAN		
05	Remote	DO	SREM		
06	Exhaust Fill Stop	DO	SBAS		
07	Air Fill Stop	DO	SBLS		
08	Bag Pair Fill Stop	DO	SBPS		
09	Analyze Modal Air	DO	SMLA		
10	Analyze Modal Exhaust	DO	SMAA		
11	Analyze Stop	DO	SANS		
12	Evacuate Stop	DO	SBES 0		
13	Purge Stop	DO	SBSS 0		
14	Heat Exchanger	DO	SWON SWOF		If set = TRUE, turns heat exchanger ON. If set = FALSE, turns heat exchanger OFF.
15	Blower	DO	SGON SGOF		If set = TRUE, turns blower ON. If set = FALSE, turns blower OFF.
16	Clear Driver Error	DO		4	
17	Clear AK Data Warning	DO		7	
01	Heat Exchanger	DI		1	If = TRUE, Heat Exchanger is ON.
01	Set Flow Rate Range	DCO	SDST n		n = Range
02	Exhaust Fill	DCO	SBAF n		n = Bag
03	Air Fill	DCO	SBLF n		n = Bag
04	Fill Bag Pair	DCO	SBPF n		n = Bag
05	Analyze Bag	DCO	SBAN n		n = Bag
06	Evacuate	DCO	SBEV n		n = Bag
07	Purge Bag	DCO	SBSP n		n = Bag
08	Leak Check Bag	DCO	SLEC n		n = Bag
09	Start Counter	DCO	SCGO n		n = Counter
10	Stop Counter	DCO	SCST n		n = Counter
11	Reset Counter	DCO	SCRS n		n = Counter
12	Sample Volume	DCO	SPVS n		b = Rate
13	Shutdown	DCO	SPDN n		n = minutes
01	Bag 1 Status	DSI		1	0=System Active, 1=Paused, 2=Standby.
02	Bag 2 Status	DSI		1	(same as above)
03	Bag 3 Status	DSI		1	(" " ")
04	Bag 4 Status	DSI		1	(" " ")
05	Bag 5 Status	DSI		1	(" " ")
06	Bag 6 Status	DSI		1	(" " ")
07	Bag 7 Status	DSI		1	(" " ")
08	Bag 8 Status	DSI		1	(" " ")
09	Overall System Status	DSI		1	(" " ")
10	Current Modal Function	DSI		1	0=None, 1=Analyzing Air, 2=Analyzing Exhaust
11	Driver Error Type	DSI		4,5	0=No errors, 1=Send Failure, 2=Receive timeout, 3=Error returned from device
12	Additional Error Code	DSI		6	Valid if Driver Error Type > 0. See Note.
13	AK Data Warning	DSI		7	0=No warnings, 1='#' character(s) detected in data
01	Curr. Std. Vol. Flow (AMDT) Nm3/min.	AI		2	
02	Entry Pressure (AMDT) kPa	AI		2	
03	Entry Temp. (AMDT) deg K	AI		2	
04	Curr. Pressure Diff. (AMDT) kPa	AI		2	
01	Send String (To Device)	TO		3	
01	Response String (From Device)	TI		3	
02	Response String (additional)	TI		3	

03	Driver Error Text	TI	4	Valid if Driver Error Type > 0
04	AK Data Warning Text	TI	7	Valid if AK Data Warning > 0

NOTES:

1) These input channel values are continuously sampled at a rate of 1 Hz by sending the ASTZ command to this device. At the present time, the sampling rate cannot be changed by the user.
2) These input channel values are continuously sampled at a rate of 1 Hz by sending the AMDT command to this device. At the present time, the sampling rate cannot be changed by the user.
3) These commands are used to send any string to this device and retrieve the response. Some of the AK protocol details are automatically handled for you. For example, 2 bytes are appended to the start of your string (STX and a "space") and one byte is added to the end or your string (ETX) before it is send to this device. Similarly, these 3 bytes (2 at the beginning and 1 at the end) are stripped from the response string that is returned via the "Response String" channels. NOTE: This device has 2 response string channels due to the fact that some CVS commands have responses of 350 or more characters. The "Response String (From Device)" channel is used to retrieve the first 255 characters of any response; if necessary, the "Response String (additional)" channel can be used to retrieve the second 255 characters of the response. The two pieces must be concatenated in the DSBasic TestPlan.
4) These channels are used by a TestPlan to check for various types of device communications errors. Typically, a limit is created and enabled that checks if the "Device Error Type" input becomes nonzero. If so, the TestPlan can handle the error based on the actual type. The "Driver Error Text" can be read and logged to a data file if desired. The driver latches the first error that occurs. Write any value to the "Clear Driver Error" channel to clear the current error and allow the next error that occurs to be latched.
5) An error dialog box is displayed whenever the "Driver Error Type" is set to 1 or 2. This essentially halts the driver (i.e. no more commands are sent to the device) until the operator responds by pressing the "Retry" or "Ignore" button. If "Ignore" is pressed and the command that failed was a sampling request (i.e. not initiated from the TestPlan), sampling for this device will be disabled for the remainder of the TestPlan. An error dialog box is NOT displayed if the "Driver Error Type" is set to 3. The TestPlan must check for this condition in a limit handler and process accordingly.
6) This channel provides additional information whenever the "Driver Error Type" becomes nonzero.

   If "Driver Error Type" is set to 1 or 2, then ...
   0 = No additional error information is available.
   1 = Timed out waiting for data.
   2 = Timed out waiting for terminator character.
   3 = Insufficient characters were received.
   4 = Serial Port not defined.

   If "Driver Error Type" is set to 3, then ...
   0 = No additional error information is available.
   1 = AK response contains "????" (Unknown or erroneous command)
   2 = AK response contains "NA" (Selected function not available)
   3 = AK response contains "BS" (System is busy)
   4 = AK response contains "OF" (System not in computer mode)
   5 = AK response contains "DF" (Data error)
   6 = AK response contains "SE" (Syntax error)
   7 = AK response contains a Nonzero Error Status. Limit handler should send an ASTF command to query error codes.
   8 = AK response is corrupted and/or invalid (e.g. <STX> character is missing)

7) These channels are used by a TestPlan to monitor for '#' characters in responses from this device because response data values can sometimes be preceded and/or replaced by '#' characters under various conditions. Typically, a limit is created and enabled that checks if the "AK Data Warning" input becomes nonzero. If so, the TestPlan can handle the warning as needed. The "AK Data Warning Text" can be read and logged to a data file if desired. The driver latches the first warning that occurs. Write any value to the "Clear AK Data Warning" channel to clear the current warning and allow the next warning that occurs to be latched.

GENERAL NOTES:

The previous notes mention two different commands that are sent to this device at a 1 Hz rate for each command. These alternating commands are sent at 500 mSec intervals such that the overall sampling rate of this device is 2 Hz; yet, the effective sampling rate of each command remains 1 Hz.

VERSION INFORMATION:

DyneSystemsCo., LLC Document Section Version 0.12 June 24, 1997

Pierburg Emissions Bench Channels

No	Name	Type	Command	Note	Comments
01	OnLine	DO	SREM K0		
02	OffLine	DO	SMAN KV		
03	Standby	DO	STBY K0		
04	Purge	DO	SSPL K0		
05	Sample	DO	SMGA K0		
06	Integrate	DO	SINT K0		
07	Leak Check	DO	SLEC K0		
08	Hangup Test	DO	SHUP K0		
09	Get Data	DO	AKON K0	1	
10	Get Integrated Data	DO	AIKO K0		
11	Get Integrated Data (Overall)	DO	AIKG K0		
12	Get Leak Check Result	DO	ALEC K0		
13	Get Hangup Test Result	DO	AHUP K0		
14	Get Measuring Ranges	DO	AEMB K0	1	
15	Sample Enable	DO		2	Set = TRUE to enable data acquisition (i.e. sampling) for this device. Set = FALSE to disable data acquisition for this device.
16	Test COM Connection	DO		17	
17	Clear Driver Error	DO		11	
18	Clear AK Data Warning	DO		14	
01	Select Cell	DCO	SPRU K0 n		n = Cell number (range 1 to N).
02	Get Channel Status	DCO	ASTZ Kn	3	n = Channel number (range 0 to N). See Note for more information.
03	Zero Gas	DCO	SNGA Kn		n = Channel number (range 0 to N).
04	Span Gas	DCO	SEGA Kn		n = Channel number (range 0 to N).
05	Get Span Gas Concentration	DCO	AKAK Kn Mm	15	n = Channel number (range 1 to N). m = Range number (1 to M).
06	Set Range	DCO	SEMB Kn Mm	16	n = Channel number (range 1 to N). m = Range number (1 to M).
01	Standby Status	DI		4	If = TRUE, in standby mode. If = FALSE, in some other mode.
02	COM Connection Test Passed	DI		17	If = TRUE, COM port connection is working.
01	Channel 1	AI		5	
02	Channel 2	AI		5	
03	Channel 3	AI		5	
04	Channel 4	AI		5	
05	Channel 5	AI		5	
06	Channel 6	AI		5	
07	Channel 7	AI		5	
08	Channel 8	AI		5	
09	Channel 1 (Integrated)	AI		7	
10	Channel 2 (Integrated)	AI		7	
11	Channel 3 (Integrated)	AI		7	
12	Channel 4 (Integrated)	AI		7	
13	Channel 5 (Integrated)	AI		7	
14	Channel 6 (Integrated)	AI		7	
15	Channel 7 (Integrated)	AI		7	
16	Channel 8 (Integrated)	AI		7	
17	Leak Check Result	AI		8	
18	Hangup Check Result	AI		9	
19	Span Gas Concentration	AI		15	
01	Channel 1 Meas. Range	DSI		6	Measuring range value (1 to N)
02	Channel 2 Meas. Range	DSI		6	(same as above)
03	Channel 3 Meas. Range	DSI		6	(" " ")
04	Channel 4 Meas. Range	DSI		6	(" " ")
05	Channel 5 Meas. Range	DSI		6	(" " ")
06	Channel 6 Meas. Range	DSI		6	(" " ")
07	Channel 7 Meas. Range	DSI		6	(" " ")
08	Channel 8 Meas. Range	DSI		6	(" " ")

09	Driver Error Type	DSI	11,12	0=No errors, 1=Send Failure, 2=Receive timeout, 3=Error returned from device
10	Additional Error Code	DSI	13	Valid if Driver Error Type > 0. See Note.
11	AK Data Warning	DSI	14	0=No warnings, 1='#' character(s) detected in data

01	Send String (To Device)	TO	10	

01	Response String (From Device)	TI	10	
02	Response String (additional)	TI	10	
03	Driver Error Text	TI	11	Valid if Driver Error Type > 0
04	AK Data Warning Text	TI	14	Valid if AK Data Warning > 0

NOTES:

1) These commands are automatically sent to the device at a rate of 1 Hz when sampling is enabled; thus, there is no reason to write anything to these channels unless sampling is disabled. These commands are alternately sent at a rate of 2 Hz which results in an effective rate of 1 Hz for each of the individual commands.
2) The ability to enable/disable sampling is a unique feature of this device driver. Some installations have multiple Cell Assistant systems sharing a single Gas Analyzer. The switching of the Gas Analyzer COM port from one Cell Assistant system to another is under the control of a PLC. Each Cell Assistant system issues request/release commands to the PLC via other PLC channels. In order to avoid COM port errors during these transitions, the TestPlan can use this channel to disable communications to the Gas Analyzer prior to releasing the Gas Analyzer. The TestPlan can re-enable communications when the Gas Analyzer is requested and granted by the PLC.
3) If you set the "Get Channel Status" channel = 0, the string "ASTZ KV" is sent to the device; otherwise, the specified channel number is put into the command string as you expect (i.e. "Get Channel Status = 3" results in the string "ASTZ K3" being sent to the device).
4) This input value is updated by sending the ASTZ command (i.e. write any value to the "Get Channel Status" channel). This causes the Pierburg to send back the current standby status of the specified channel.
5) When sampling is enabled, these input values are continuously sampled at a rate of 1 Hz by sending the AKON K0 command to this device. At the present time, the sampling rate cannot be changed by the user. When sampling is disabled, these input values can be updated manually by sending the AKON K0 command (i.e. write any value to the "Get Data" channel).
6) When sampling is enabled, these input values are continuously sampled at a rate of 1 Hz by sending the AEMB K0 command to this device. At the present time, the sampling rate cannot be changed by the user. When sampling is disabled, these input values can be updated manually by sending the AEMB K0 command (i.e. write any value to the "Get Measuring Ranges" channel).
7) These input values are updated by sending the AIKO K0 or AIKG K0 command (i.e. write any value to the "Get Integrated Data" or "Get Integrated Data (Overall)" channels respectively). The TestPlan writer must keep track of which of the two commands was sent most recently in order to properly interpret the meaning of these integrated data channels.
8) This input value is updated by sending the ALEC K0 command (I.e. write any value to the "Get Leak Check Result" channel).
9) This input value is updated by sending the AHUP K0 command (I.e. write any value to the "Get Hangup Test Result" channel).
10) These commands are used to send any string to this device and retrieve the response. Some of the AK protocol details are automatically handled for you. For example, 2 bytes are appended to the start of your string (STX and a "space") and one byte is added to the end or your string (ETX) before it is send to this device. Similarly, these 3 bytes (2 at the beginning and 1 at the end) are stripped from the response string that is returned via the "Response String" channels. NOTE: This device has 2 response string channels due to the fact that some bench commands have responses of 350 or more characters. The "Response String (From Device)" channel is used to retrieve the first 255 characters of any response; if necessary, the "Response String (additional)" channel can be used to retrieve the second 255 characters of the response. The two pieces must be concatenated in the DSBasic TestPlan.
11) These channels are used by a TestPlan to check for various types of device communications errors. Typically, a limit is created and enabled that checks if the "Device Error Type" input becomes nonzero. If so, the TestPlan can handle the error based on the actual type. The "Driver Error Text" can be read and logged to a data file if desired. The driver latches the first error that occurs. Write any value to the "Clear Driver Error" channel to clear the current error and allow the next error that occurs to be latched.
12) An error dialog box is displayed whenever the "Driver Error Type" is set to 1 or 2. This essentially halts the driver (i.e. no more commands are sent to the device) until the operator responds by pressing the "Retry" or "Ignore" button. If "Ignore" is pressed and the command that failed was a sampling request (i.e. not initiated from the TestPlan), sampling for this device will be disabled for the remainder of the TestPlan. An error dialog box is NOT displayed if the "Driver Error Type" is set to 3. The TestPlan must check for this condition in a limit handler and process accordingly.
13) This channel provides additional information whenever the "Driver Error Type" becomes nonzero.

If "Driver Error Type" is set to 1 or 2, then ...

0 = No additional error information is available.
1 = Timed out waiting for data.
2 = Timed out waiting for terminator character.
3 = Insufficient characters were received.
4 = Serial Port not defined.

If "Driver Error Type" is set to 3, then ...

0 = No additional error information is available.
1 = AK response contains "????" (Unknown or erroneous command)
2 = AK response contains "NA" (Selected function not available)
3 = AK response contains "BS" (System is busy)
4 = AK response contains "OF" (System not in computer mode)
5 = AK response contains "DF" (Data error)
6 = AK response contains "SE" (Syntax error)
7 = AK response contains a Nonzero Error Status. Limit handler should send an ASTF command to query error codes.
8 = AK response is corrupted and/or invalid (e.g. <STX> character is missing)

14) These channels are used by a TestPlan to monitor for '#' characters in responses from this device because response data values can sometimes be preceded and/or replaced by '#' characters under various conditions. Typically, a limit is created and enabled that checks if the "AK Data Warning" input becomes nonzero. If so, the TestPlan can handle the warning as needed. The "AK Data Warning Text" can be read and logged to a data file if desired. The driver latches the first warning that occurs. Write any value to the "Clear AK Data Warning" channel to clear the current warning and allow the next warning that occurs to be latched.
15) The data values that you write to this channel are in the form of "nm" (i.e. the ten's digit is the channel number and the 1's digit is the range number). For example, to get the concentration of channel 2 range 3, you would write a 23 to the "Get Span Gas Concentration" channel. Afterwards, the result is read back by reading the "Span Gas Concentration" channel.
16) The data values that you write to this channel are in the form of "nm" (i.e. the ten's digit is the channel number and the 1's digit is the range number). For example, to set the range of channel 3 to range 1, you would write a 31 to the "Set Range" channel.
17) Writing anything to the "Test COM Connection" channel causes the driver to send a harmless command (ASTF) to this device. The driver then checks that a valid response is received. The results of this test can be determined by reading the "COM Connection Test Passed" channel (TRUE = passed, FALSE = failed). The "Test COM Connection" channel is mainly intended to be used in systems that multiplex the Emissions Bench COM port between two or more host computers. After each switch of the multiplexor, the "Test COM Connection" channel should be used to flush out any spurious characters that may have been generated by the switching operation. If the COM connection is NOT flushed after each switch of the multiplexor, the first command that is sent to the Emissions Bench after the switch may result in a communications error. NOTE: The "Test COM Connection" channel should only be used when sampling is NOT enabled for this device; otherwise, the results are unpredictable. Of coarse, the COM port multiplexor should never be switched when sampling is enabled anyway, so this should not be a problem.

GENERAL NOTES:
The previous notes mention two different commands that are sent to this device at a 1 Hz rate for each command. These alternating commands are sent at 500 mSec intervals such that the overall sampling rate of this device is 2 Hz; yet, the effective sampling rate of each command remains 1 Hz.

VERSION INFORMATION:
DyneSystemsCo., LLC Document Section Version 0.14 August 29, 1997

Rosemount Micro Motion RFT9712 Fuel Meter Channels

No	Name	Type	Command	Note	Comments
01	Start Totalizer	DO			
02	Stop Totalizer	DO			
03	Reset Totalizer	DO			
04	Clear Driver Error	DO		2	
01	Flow Rate	AI		1,5	See Note 5 for information about channel units.
02	Temperature	AI		1,5	
03	Totalizer	AI		1,5	
04	Density	AI		1,5	
01	Driver Error Type	DSI		2,3	0=No errors, 1=Send Failure, 2=Receive timeout, 3=Error returned from device
02	Additional Error Code	DSI		4	Valid if Driver Error Type > 0. See Note.
01	Driver Error Text	TI		2	Valid if Driver Error Type > 0

NOTES:

1) These input channel values are continuously sampled at a rate of 1Hz.
2) These channels are used by a TestPlan to check for various types of device communications errors. Typically, a limit is created and enabled that checks if the "Device Error Type" input becomes nonzero. If so, the TestPlan can handle the error based on the actual type. The "Driver Error Text" can be read and logged to a data file if desired. The driver latches the first error that occurs. Write any value to the "Clear Driver Error" channel to clear the current error and allow the next error that occurs to be latched.
3) An error dialog box is displayed whenever the "Driver Error Type" becomes nonzero. This essentially halts the driver (i.e. no more commands are sent to the device) until the operator responds by pressing the "Retry" or "Ignore" button. If "Ignore" is pressed and the command that failed was a sampling request (i.e. not initiated from the TestPlan), sampling for this device will be disabled for the remainder of the TestPlan.
4) This channel provides additional information whenever the "Driver Error Type" becomes nonzero.

   If "Driver Error Type" is set to 1 or 2, then ...
   0 = No additional error information is available.
   1 = Timed out waiting for data.
   2 = Timed out waiting for terminator character.
   3 = Insufficient characters were received.
   4 = Serial Port not defined.

   If "Driver Error Type" is set to 3, then ...
   Always = 0. No additional error information is available.

5) The "units" for each of the analog input channels (except the "Totalizer" channel) is configurable from the "Settings" page for this driver. The Fuel Meter automatically selects the most appropriate units for the "Totalizer" channel based on the unit selections for the other channels. The currently active unit selections can be read back and displayed by pressing the "Data" button on the "Diagnostic" page for this driver.

GENERAL NOTES:
None.

VERSION INFORMATION:
DyneSystemsCo., LLC Document Section Version 0.01 June 25, 1999

Sartorius LC3201D Channels

No	Name	Type	Command	Note	Comments
01	Tare	DO	<ESC>T<CR><LF>		
02	Reset Totalizer 1	DO			
03	Reset Totalizer 2	DO			
04	Reset Totalizer 3	DO			
05	Reset All Totalizers	DO		5	
06	Clear Driver Error	DO		2	
01	Reading Is Stable	DI		1	Reading = the “Current Weight (grams)” channel.
02	Above Full Level	DI		1	
03	Below Empty Level	DI		1	
04	Overload	DI		1	
05	Underload	DI		1	
01	Current Weight (grams)	AI		1	
02	Totalizer 1 Rate (grams/min)	AI			
03	Totalizer 2 Rate (grams/min)	AI			
04	Totalizer 3 Rate (grams/min)	AI			
05	Totalizer 1 Measurement Interval (min)	AI			
06	Totalizer 2 Measurement Interval (min)	AI			
07	Totalizer 3 Measurement Interval (min)	AI			
01	Driver Error Type	DSI		2,3	0=No errors, 1=Send Failure, 2=Receive timeout, 3=Error returned from device
02	Additional Error Code	DSI		4	Valid if Driver Error Type > 0. See Note.
03	IAC Error Code	DSI		4	
01	Send Control Command	TO	<ESC>xxx<CR><LF>	6	“xxx” is any text string
01	Driver Error Text	TI		2	Valid if Driver Error Type > 0

NOTES:

1) These input channel values are continuously sampled at a rate of 1 Hz. The device driver sends a “Print” control command to the device to obtain each reading. The print command sequence is <ESC>P<CR><LF>.
2) These channels are used by a TestPlan to check for various types of device communications errors. Typically, a limit is created and enabled that checks if the “Device Error Type” input becomes nonzero. If so, the TestPlan can handle the error based on the actual type. The “Driver Error Text” can be read and logged to a data file if desired. The driver latches the first error that occurs. Write any value to the “Clear Driver Error” channel to clear the current error and allow the next error that occurs to be latched. This channel also clears the “IAC Error Code” channel.
3) An error dialog box is displayed whenever the “Driver Error Type” becomes nonzero. This essentially halts the driver (i.e. no more commands are sent to the device) until the operator responds by pressing the “Retry” or “Ignore” button. If “Ignore” is pressed and the command that failed was a sampling request (i.e. not initiated from the TestPlan), sampling for this device will be disabled for the remainder of the TestPlan.
4) This channel provides additional information whenever the “Driver Error Type” becomes nonzero.

   If “Driver Error Type” is set to 1 or 2, then ...
   0 = No additional error information is available.
   1 = Timed out waiting for data.
   2 = Timed out waiting for terminator character.
   3 = Insufficient characters were received.
   4 = Serial Port not defined.

   If “Driver Error Type” is set to 3, then ...
   0 = No additional error information is available.
   1 = Device is not reading in GRAMS.
   2 = IAC Error code returned from device. (Read “IAC Error Code” channel to get actual code)

5) Use this channel if you want all Totalizers to be reset at “exactly” the same time.
6) The Sartorius manual defines a set of “Control Commands” that can be sent to this device in order to modify various device codes. Keypad operation can also be simulated. This channel will encapsulate any ASCII string (i.e. a Control Command) with an <ESC> prefix character and a <CR><LF> suffix pair and send it to this device. This channel may be helpful in advanced programming situations and should generally not be used in most applications. In fact, altering some device codes may cause this device driver to become unreliable or totally non-functional since the driver requires some device codes to remain constant.

GENERAL NOTES:
None.

VERSION INFORMATION:
DyneSystemsCo., LLC Document Section Version 0.03 April 16, 1997

Symbol Technologies LS1004 Channels

No	Name	Type	Command	Note	Comments
01	Clear Previous Scan	DO		1	
01	Scan Data Available	DI		2	
01	Scan Data	TI		3	Only valid if “Scan Data Available” = TRUE.

NOTES:

1) The currently running TestPlan must always write a value to this channel to clear the previous scan and prepare the driver to receive another scan. The previous scan remains latched in the driver until it is cleared using this channel.
2) The currently running TestPlan should periodically read this channel to detect the availability of a new scan. Reading this channel causes the driver to process all scanned characters received so far. When a complete scan is received, this channel will return a TRUE value; othewise, it will return a FALSE value. Once a scanned value is available and this channel is set to TRUE, it can only be cleared using the “Clear Previous Scan” channel described above.
3) This channel returns the last complete scanned value.

GENERAL NOTES:

This is NOT a sampled device. The currently running TestPlan must control when an operator uses the scanner and then poll the scanner using the “Scan Data Available” channel as described above. The value of the “Scan Data” channel is only valid after the “Scan Data Channel” channel returns a value of TRUE.

VERSION INFORMATION:

DyneSystemsCo., LLC Document Section Version 0.01 March 2, 1999

SYSTEM Channels

No	Name	Type	Command	Note	Comments
01	Clear Watchdog Incident	DO		1	
02	Force Watchdog Incident	DO		1	Allows TestPlan to trip the watchdog hardware.
01	TestPlan Calc Overrun	DI		3	
02	Data File Overrun	DI		4	
03	Data File Overrun (Latched)	DI		4	
01	Watchdog Incident Code	DSI		1,2	0=No incidents. (See note 2 for additional codes)
01	Watchdog Incident Source	TI		1	
02	Watchdog Incident Description	TI		1	

NOTES:

1) These channels allow a running TestPlan to interface with the watchdog mechanism. Typically, a limit is created and enabled that checks if the "Watchdog Incident Code" channel becomes nonzero. If so, the TestPlan can handle the incident based on the actual code. The "Watchdog Incident Source" and "Watchdog Incident Description" channels can be read and logged in a data file if needed. Write any value to the "Clear Watchdog Incident" channel to clear the current incident and allow another (if any) to be logged.
2) This channel will always be set to one of the following values.
   0 = No watchdog incidents detected.
   1 = Watchdog hardware tripped due to internal software error.
   2 = Watchdog hardware tripped because a Device Communications error occurred (if enabled, see below).
   3 = Watchdog hardware tripped because a TestPlan Runtime error occurred (if enabled, see below).
   4 = Watchdog hardware tripped because the TestPlan forced the trip.

   Some of the incident codes can be disabled in the Cell Assistant preferences pages. Disabling an incident code merely permits the watchdog mechanism to keep updating the external watchdog hardware (i.e. keep it alive) even though the incident occurred. The incident code will still be set such that an enabled limit handler can still detect and process the incident as needed.

3) Cell Assistant updates all virtual channels at 20Hz in a highly synchronized fashion. TestPlan calculated channels take a disproportionately long time to compute because they require the DSBasic engine to interrupt the currently running TestPlan in order to execute the DSBasic code for each function. Certain DSBasic code functions (in the running TestPlan) are NOT interruptible which may significantly delay computation of the TestPlan calculated functions. In order to not destroy the tight synchronization of all other channels, the computation of TestPlan calculated channels will be delayed in these instances and the "TestPlan Calc Overrun" channel will be set to TRUE. It will be set back to FALSE as soon as the TestPlan calculated channels resynchronize with the rest of the system.
4) Logging to data files normally occurs with no errors. Effective with Release 1.3, data files now grant READ share permission to other applications even while they are opened and being written to by a running TestPlan. Even though Cell Assistant has exclusive WRITE permission and other applications only have READ access permission, Cell Assistant can still be denied access to the file during the "instant" that another application reads in the contents of the data file. This "instant" can range from milliseconds to several seconds depending on the size of the file. During this time, it is possible that some logged data will be lost. If this occurs, the "Data File Overrun" channel will be set to TRUE. It will be set to FALSE as soon as file access becomes available again and Cell Assistant resumes data logging. The "Data File Overrun (Latched)" channel will also be set to TRUE when an overrun occurs; however, it will only be cleared when the TestPlan stops.

GENERAL NOTES:

The SYSTEM device is not a physical device at all; instead, it is a pseudo-device that exists in all TestSystems. You cannot add it to or remove it from a TestSystem (as you can with other hardware devices). It is simply always present. SYSTEM channels provide a communications mechanism between any running TestPlan and various Cell Assistant internal state variables.

VERSION INFORMATION:

DyneSystemsCo., LLC Document Section Version 0.02 April 8, 1998

Virtual Instrument Panel (VIP) Driver Channels

No	Name	Type	Command	Note	Comments
001	Digital Channel 01	DO			
002	Digital Channel 02	DO			
...	etc ...	DO			
255	Digital Channel 255	DO			
256	Digital Channel 256	DO			
001	Analog Channel 01	AO			
002	Analog Channel 02	AO			
...	etc ...	AO			
255	Analog Channel 255	AO			
256	Analog Channel 256	AO			
001	Text Channel 01	TO			
002	Text Channel 02	TO			
...	etc ...	TO			
255	Text Channel 255	TO			
256	Text Channel 256	TO			
001	Device Control Channel 01	DCO			
002	Device Control Channel 02	DCO			
...	etc ...	DCO			
255	Device Control Channel 255	DCO			
256	Device Control Channel 256	DCO			

NOTES:
None.

GENERAL NOTES:
The Virtual Instrument Panel (VIP) driver is not actually connected to a physical device (like the other device drivers are). This driver exists in order to create channels that can be used to exchange data between a TestPlan and a Virtual Instrument Panel executable that has been created using Microsoft Visual Basic. Usually, a VIP executable monitors data that is exchanged between a TestPlan and a real device driver; it does this using the CADAQ OLE control. This control gives a VIP executable access to all data in all channels. In some cases, however, a TestPlan may need to exchange data directly with a VIP executable (i.e. display a message, illuminate an LED, etc ...). The VIP driver provides a large set of channels (of all types) for exactly this purpose. Data can be exchanged with the VIP driver (and hence the VIP executable) without actually exchanging any data with any physical device.

Notice that all VIP driver channels are “output” channels. This is only true in the Cell Assistant sense; that is, they can only be created by creating a channel of type “Digital Output”, “Analog Output”, “Text Output”, or “Device Control (output)” in Cell Assistant for Windows. But since “output” channels can also be “read” by a TestPlan, any of these channels can be used to write data to or read data from a VIP executable after they have been created.

VERSION INFORMATION:
DyneSystemsCo., LLC Document Section Version 0.03 May 3, 2001

XM Series Gas Analyzer Channels

No	Name	Type	Command	Note	Comments
01	System Reset	DO	SRES	9	
02	Sample Subsystem Reset	DO	ECSS	9	
03	Standby	DO	STBY		
04	Flow Zero Gas	DO	SNGA		
05	Flow Sample Gas	DO	SMGA		
06	Purge	DO	SSPL		
07	Hangup Test	DO	SHUP		
08	Leak Check	DO	SLEC		
09	Autozero Analyzer	DO	SAZR		
10	Get Hangup Test Result	DO	AHUP		
11	Get Leak Check Result	DO	ALEC		
12	Sample Enable	DO		7	
13	Clear Driver Error	DO		2	
01	Filter Differential Pressure Warning	DI		1	TRUE = pressure is too high.
02	Zero Calibration Requested	DI		1	
03	New NOx Sensor Required	DI		1	
04	New O2 Sensor Required	DI		1	
01	Flow Span Gas	DCO	SEGA Gn		n = span gas port (range 1 thru 5).
02	Autospan Gas	DCO	SASP Gn	10	n = span gas port (range 0 thru 5).
03	Autospan Specific Gas	DCO	SASP Kn		n = gas (1=NO, 2=HC, 3=CO2, 4=CO, 5=O2).
01	Concentration – NO	AI		1	
02	Concentration – HC	AI		1	
03	Concentration – CO2	AI		1	
04	Concentration – CO	AI		1	
05	Concentration – O2	AI		1	
06	Hangup Test Result	AI		5	
07	Leak Check Result	AI		6	
01	Analyzer Mode	DSI		1	1=STBY, 2=SNGA, 3=SEGA, 4=SMGA, 5=SSPL, 6=SHUP, 7=SLEC, 8=SAZR, 9=SASP, 10=SRES, or 0=In transition between modes.
02	Sample Path Vacuum Status	DSI		1	0=Normal, 1=Warning level exceeded, 2=Failure level exceeded
03	Andros Status Bytes	DSI		1,8	See note 9.
04	Driver Error Type	DSI		2,3	0=No errors, 1=Send Failure, 2=Receive timeout, 3=Error returned from device
05	Additional Error Code	DSI		4	Valid if Driver Error Type > 0. See Note.
01	Driver Error Text	TI		2	Valid if Driver Error Type > 0

NOTES:

1) These input channel values are continuously sampled at a rate of 1 Hz using the ANDR command.
2) These channels are used by a TestPlan to check for various types of device communications errors. Typically, a limit is created and enabled that checks if the "Device Error Type" input becomes nonzero. If so, the TestPlan can handle the error based on the actual type. The "Driver Error Text" can be read and logged to a data file if desired. The driver latches the first error that occurs. Write any value to the "Clear Driver Error" channel to clear the current error and allow the next error that occurs to be latched.
3) An error dialog box is displayed whenever the "Driver Error Type" becomes nonzero. This essentially halts the driver (i.e. no more commands are sent to the device) until the operator responds by pressing the "Retry" or "Ignore" button. If "Ignore" is pressed and the command that failed was a sampling request (i.e. not initiated from the TestPlan), sampling for this device will be disabled for the remainder of the TestPlan.
4) This channel provides additional information whenever the "Driver Error Type" becomes nonzero.

If "Driver Error Type" is set to 1 or 2, then ...

0 = No additional error information is available.
1 = Timed out waiting for data.
2 = Timed out waiting for terminator character.
3 = Insufficient characters were received.
4 = Serial Port not defined.

If "Driver Error Type" is set to 3, then ...
Always = 0 (No additional error information is available at this time).

5) This input value is updated by sending the AHUP command to this device (i.e. write any value to the "Get Hangup Test Result" channel).
6) This input value is updated by sending the ALEC command to this device (i.e. write any value to the "Get Leak Check Result" channel).
7) This channel is used to enable/disable data acquisition (i.e. sampling) for this device. This allows a running TestPlan to control when and how often this device is sampled.
8) The XM Series Gas Analyzer is currently based on the Andros Model 6600 Miniature Automotive Gas Analyzer. The Andros provides four 8-bit bytes of status information that are read and returned in the response to the AKON command (see Note 1). Some of the status bit information in these bytes is automatically extracted and presented as digital input channels to Cell Assistant (e.g. the "Zero Calibration Requested" channel). The "Andros Status Bytes" channel packs the four 8-bit Andros status bytes into one 32-bit status word and presents it to Cell Assistant as a single Device Status Input (DSI) channel. Bits 0-7 are Andros status byte 4, bits 8-15 are Andros status byte 3, bits 16-23 are Andros status byte 2, and bits 24-31 are Andros status byte 1. See the Andros manual for the definition of all status bits.
9) The "System Reset" channel will completely reset the XM Series Gas Analyzer. A complete recalibration of the analyzer will then be required. If a recalibration is undesirable, the "Sample Subsystem Reset" channel should be used to clear most errors that occur in the XM Series Gas Analyzer.
10) Span Gas ports 1 thru 5 can contain all types of span gases. Port 0 typically contains a "zero" gas; however, it can also contain a span gas for oxygen. No other span gases should exist in the gas container that is connected to port 0.

GENERAL NOTES:
None.

VERSION INFORMATION:
DyneSystemsCo., LLC Document Section Version 0.08 October 20, 2000

Appendix B - DSBasic Language Extensions

TestPlanSleep (function)

Syntax `TestPlanSleep` *number*
Description **Causes the TestPlan to stop running for the specified number of milliseconds.**
Comments **The value of *number* must be a `Long` greater than 0.**
Example
See Also

SuspendWatchdog (function)

Syntax `SuspendWatchdog` *number*
Description Causes the watchdog thread to stop monitoring other threads for the specified number of seconds.
Comments The value of *number* must be a `Long` in the range of 1 to 20 seconds (inclusive) Some DSBasic language commands (e.g. OLE automation) take a long time to execute which may cause other threads to block and subsequently initiate nuisance watchdog trips. This command can be used prior these lengthy DSBasic commands in order to keep the watchdog alive for the duration.
Example
See Also

CurrentLimitSet (property)

Syntax
```
CurrentLimitSet = "LimitSetName"
StringValue = CurrentLimitSet
```
Description The first syntax example causes the currently running TestPlan to switch to the specified limit set. The second syntax example retrieves the name of the currently selected limit set.
Comments If the specified *LimitSetName* is not valid, a DSBasic runtime error will occur.
Example
See Also

Limit.Channel (method)

Syntax *text$* `= Limit.Channel`
Description Returns the name of the channel that is being monitored by the limit.
Comments
Example
See Also `Limit.Disable` (method), `Limit.Enable` (method), `Limit.Function` (method), `Limit.High` (method), `Limit.Low` (method), `Limit.Priority` (method), `Limit.Status` (method)

Limit.Disable (method)

Syntax `Limit.Disable`

Description Disables the limit object so that this limit will no longer be checked.
Comments
Example
See Also `Limit.Channel` (method), `Limit.Enable` (method), `Limit.Function` (method), `Limit.High` (method), `Limit.Low` (method), `Limit.Priority` (method), `Limit.Status` (method)

Limit.Enable (method)

Syntax `Limit.Enable`
Description Enables the limit object so that this limit will be checked.
Comments When a TestPlan starts to run all limits are disabled. This method must be used for each limit that needs to be checked while the TestPlan is running.
Example
See Also `Limit.Channel` (method), `Limit.Disable` (method) `Limit.Function` (method), `Limit.High` (method), `Limit.Low` (method), `Limit.Priority` (method), `Limit.Status` (method)

Limit.Function (method)

Syntax *text$* `= Limit.Function`
Description Returns the name of the limit procedure which will run if the limit object exceeds it bounds.
Comments
Example
See Also `Limit.Channel` (method), `Limit.Disable` (method), `Limit.Enable` (method), `Limit.High` (method), `Limit.Low` (method), `Limit.Priority` (method), `Limit.Status` (method)

Limit.High (method)

Syntax *number* `= Limit.High`
`Limit.High` *number*
Description Used to set or get the High Limit value for a limit object. The value returned (or used as a parameter) is a `Single`.
Comments
Example
See Also `Limit.Channel` (method), `Limit.Disable` (method), `Limit.Enable` (method), `Limit.Function` (method), `Limit.Low` (method), `Limit.Priority` (method), `Limit.Status` (method)

Limit.IsTripped (method)

Syntax *boolean* `= Limit.IsTripped`
Description Returns a `Boolean` describing the trip state of this limit. The trip state is the result of the last time the limit thread evaluated this limit. Trip conditions may have changed since then. Use the `Limit.Status` method to check if trip conditions currently exist.
Comments
Example

See Also `Limit.Channel` (method), `Limit.Disable` (method), `Limit.Enable` (method), `Limit.Function` (method), `Limit.High` (method), `Limit.Low` (method), `Limit.Priority` (method)

Limit.Low (method)

Syntax *number* = `Limit.Low`
`Limit.Low` *number*

Description Used to set or get the Low Limit value for a limit object. The value returned (or used as a parameter) is a `Single`.

Comments

Example

See Also `Limit.Channel` (method), `Limit.Disable` (method), `Limit.Enable` (method), `Limit.Function` (method), `Limit.High` (method), `Limit.Priority` (method), `Limit.Status` (method)

Limit.Priority (method)

Syntax *boolean* = `Limit.Priority`

Description Returns a `Boolean` indicating whether or not this limit object is High priority or not.

Comments

Example

See Also `Limit.Channel` (method), `Limit.Disable` (method), `Limit.Enable` (method), `Limit.Function` (method), `Limit.High` (method), `Limit.Low` (method), `Limit.Status` (method)

Limit.Status (method)

Syntax *boolean* = `Limit.Status`

Description Returns a `Boolean` describing whether or not trip conditions currently exist for this limit. The limit is not actually evaluated (i.e. the current trip state of the limit is not updated).

Comments

Example

See Also `Limit.Channel` (method), `Limit.Disable` (method), `Limit.Enable` (method), `Limit.Function` (method), `Limit.High` (method), `Limit.Low` (method), `Limit.Priority` (method)

Channel.Autozero (method)

Syntax `Channel.Autozero`

Description Sets the offset value for an analog input channel to be the inverse of the current value of the channel.

Comments This method is only valid for analog input channels. Using it on an analog output channel will have no effect. Using it on any other type of channel will cause a compiler error.

The affect of calling Autozero while measuring a non-zero value is to cancel that value out of the channel measurement. Since this method affects the offset attribute of the channel, the affect is only maintained while the TestPlan is running. When the

TestPlan stops running, the Offset attribute of the channel is returned to its previous value.

Example

See Also `Channel.Preset` (method), `Channel.Reset` (method), `Channel.Start` (method), `Channel.Stop` (method), `Channel.Text` (method), `Channel.Units` (method), `Channel.Value` (method)

Channel.Preset (method)

Syntax `Channel.Preset [`*`value`*`]`

Description Sets the value of a counter or timer channel to a preset value. The channel will be set to the specified *value* if present; if the *value* is omitted, the channel will be set to the default preset value that was specified in the channel setup dialog box.

Comments This method is only valid for counter and timer channels. Using it on any other type of channel will cause a compiler error.

Example

See Also `Channel.Autozero` (method), `Channel.Reset` (method), `Channel.Start` (method), `Channel.Stop` (method), `Channel.Text` (method), `Channel.Units` (method), `Channel.Value` (method)

Channel.Reset (method)

Syntax `Channel.Reset`

Description Resets a counter, timer, or average channel.

Comments This method is only valid for counter, timer, and average channels. Using it on any other type of channel will cause a compiler error.

Using this method for a counter or timer channel will cause the channels value to be reset to zero. Using this method for an average channel will cause the averaging to start over based on the current value of the channel that is being averaged.

Example

See Also `Channel.Autozero` (method), `Channel.Preset` (method), `Channel.Start` (method), `Channel.Stop` (method), `Channel.Text` (method), `Channel.Units` (method), `Channel.Value` (method)

Channel.Start (method)

Syntax `Channel.Start`

Description Starts counting state changes on the input to a counter channel. For timer channels, this method allows the timer to begin incrementing or decrementing (as defined in the timer channel setup dialog box) at a rate of 1 Hz.

Comments This method is only valid for counter and timer channels. Using it on any other type of channel will cause a compiler error.

Example

See Also `Channel.Autozero` (method), `Channel.Preset` (method), `Channel.Reset` (method), `Channel.Stop` (method), `Channel.Text` (method), `Channel.Units` (method), `Channel.Value` (method)

Channel.Stop (method)

Syntax `Channel.Stop`

Description Stops counting state changes on the input to a counter channel. Causes timer channels to stop also.

Comments This method is only valid for counter and timer channels. Using it on any other type of channel will cause a compiler error.

Example

See Also `Channel.Autozero` (method), `Channel.Preset` (method), `Channel.Reset` (method), `Channel.Start` (method), `Channel.Text` (method), `Channel.Units` (method), `Channel.Value` (method)

Channel.Text (method)

Syntax *text$* `= Channel.Text`

Description Returns a `String` associated with the current state of a digital channel.

Comments A digital channel can have a text string associated with each of its states. These strings are configured in the channel modify dialog. If no text has been entered for the current state of the channel, this routine will return an empty string.

Example

See Also `Channel.Autozero` (method), `Channel.Preset` (method), `Channel.Reset` (method), `Channel.Start` (method), `Channel.Stop` (method), `Channel.Text` (method), `Channel.Units` (method), `Channel.Value` (method)

Channel.Units (method)

Syntax *text$* `= Channel.Units`

Description Returns a `String` with the units for this channel.

Comments A channel's units are configured in the channel modify dialog. If no units have been entered for a channel, this routine will return an empty string.

Example

See Also `Channel.Autozero` (method), `Channel.Preset` (method), `Channel.Reset` (method), `Channel.Start` (method), `Channel.Stop` (method), `Channel.Text` (method), `Channel.Value` (method)

Channel.Value (method)

Syntax *text$ or number* `= Channel.Value`
`Channel.Value` *text$ or number*

Description Sets or gets the value of a channel.

Comments The value of a channel can be set or retrieve using this method. However, the behavior of the method will change depending on the type and direction of the channel. Certain combinations of channel type and direction are not currently supported. Below are two tables describing the behavior of this method depending on channel type, channel direction, and getting or setting the value.

Getting the value of a channel

Type	Direction	Return Type
Analog	Input	Double
Analog	Output	Double
Average	Input	Double
Calculated	Input	Double
Counter	Input	Double
DDE Input	Input	String
Device Status	Input	Double
Device Control	Output	Double
Digital	Input	Boolean
Digital	Output	Boolean
Text	Input	String
Text	Output	String
Timer	Input	Double

Setting the value of a channel

Type	Direction	Input Type	Comment
Analog	Input	Double	Not Supported
Analog	Output	Double	
Average	Input	Double	Not Supported
Calculated	Input	Double	Not Supported
Counter	Input	Double	Not Supported
DDE Input	Input	String	Not Supported
Device Status	Input	Double	Not Supported
Device Control	Output	Double	
Digital	Input	Boolean	Not Supported
Digital	Output	Boolean	
Text	Input	String	Not Supported
Text	Output	String	
Timer	Input	Double	Not Supported

This method is the default method for a channel object. This means that using the channel object name without the `.Value` method name will have the same effect.

Example

See Also `Channel.Autozero` (method), `Channel.Preset` (method), `Channel.Reset` (method), `Channel.Start` (method), `Channel.Stop` (method), `Channel.Text` (method), `Channel.Units` (method)

DataFileFormat.Close (method)

Syntax `DataFileFormat.Close [`*noheaderflag*`]`

Description Used to close a data file object.

Comments If the *noheaderflag* is omitted, a text entry is added to the data file indicating the data file has been closed; if the *noheaderflag* is set to `NoHeader` or `NH`, no message is added to the data file.

Example

See Also `DataFileFormat.Copy` (method), `DataFileFormat.FileUpdate` (method), `DataFileFormat.ForceEntry` (method), `DataFileFormat.IsOpen` (method), `DataFileFormat.Lograte` (method), `DataFileFormat.Open` (method), `DataFileFormat.Snapshot` (method), `DataFileFormat.StartLogging` (method), `DataFileFormat.StopLogging` (method) , `DataFileFormat.WriteChannelHeader` (method), `DataFileFormat.WriteChannelNames` (method), `DataFileFormat.WriteChannelUnits` (method)

DataFileFormat.Copy (method)

Syntax `DataFileFormat.Copy [`*noheaderflag*`]`

Description Used to make a copy of the current contents of the data file. The copied file will be named "xxxCopy.yyy" where "xxx.yyy" was the name of the original data file. It will be placed in the same directory as the original file.

Comments If the *noheaderflag* is omitted, a text entry is added to the data file indicating the data file has been copied; if the *noheaderflag* is set to `NoHeader` or `NH`, no message is added to the data file.

Example

See Also `DataFileFormat.Close` (method), `DataFileFormat.FileUpdate` (method), `DataFileFormat.ForceEntry` (method), `DataFileFormat.IsOpen` (method), `DataFileFormat.Lograte` (method), `DataFileFormat.Open` (method), `DataFileFormat.Snapshot` (method), `DataFileFormat.StartLogging` (method), `DataFileFormat.StopLogging` (method) , `DataFileFormat.WriteChannelHeader` (method), `DataFileFormat.WriteChannelNames` (method), `DataFileFormat.WriteChannelUnits` (method)

DataFileFormat.FileUpdate (method)

Syntax `DataFileFormat.FileUpdate`

Description Force all previously logged data to be written to disk and update the disk directory with the new file size. This makes the intermediate contents of a data file available to other applications.

Comments The data file must be open before this method can be called; otherwise, a runtime error will be displayed.

Example

See Also `DataFileFormat.Copy` (method), `DataFileFormat.Close` (method), `DataFileFormat.IsOpen` (method), `DataFileFormat.Lograte` (method), `DataFileFormat.Open` (method), `DataFileFormat.Snapshot` (method), `DataFileFormat.StartLogging` (method), `DataFileFormat.StopLogging` (method) , `DataFileFormat.WriteChannelHeader` (method), `DataFileFormat.WriteChannelNames` (method), `DataFileFormat.WriteChannelUnits` (method)

DataFileFormat.ForceEntry (method)

Syntax `DataFileFormat.ForceEntry` *text$*
Description Used to insert a line of text into a data file.
Comments The data file must be open before this method can be called. A runtime error will be displayed if the entry cannot be added to the data file.
Example
See Also `DataFileFormat.Close` (method), `DataFileFormat.Copy` (method), `DataFileFormat.FileUpdate` (method), `DataFileFormat.IsOpen` (method), `DataFileFormat.Lograte` (method), `DataFileFormat.Open` (method), `DataFileFormat.Snapshot` (method), `DataFileFormat.StartLogging` (method), `DataFileFormat.StopLogging` (method) , `DataFileFormat.WriteChannelHeader` (method), `DataFileFormat.WriteChannelNames` (method), `DataFileFormat.WriteChannelUnits` (method)

DataFileFormat.IsOpen (method)

Syntax `DataFileFormat.IsOpen`
Description Used to check if a data file is currently open. Returns TRUE if so; else, returns FALSE.
Comments Useful in shutdown and error handling routines that perform cleanup after an error. Checks if a data file is currently open so it can be closed if so. (Not very useful anymore since closing a data file that has not been opened no longer causes a DSBasic runtime error.)
Example
See Also `DataFileFormat.Close` (method), `DataFileFormat.Copy` (method), `DataFileFormat.FileUpdate` (method), `DataFileFormat.ForceEntry` (method), `DataFileFormat.Lograte` (method), `DataFileFormat.Open` (method), `DataFileFormat.Snapshot` (method), `DataFileFormat.StartLogging` (method), `DataFileFormat.StopLogging` (method) , `DataFileFormat.WriteChannelHeader` (method), `DataFileFormat.WriteChannelNames` (method), `DataFileFormat.WriteChannelUnits` (method)

DataFileFormat.Lograte (method)

Syntax `DataFileFormat.Lograte` *number*
Description Used to set or change the rate at which the data file logs data.
Comments The input parameter is a `Double`. Legal values for Lograte are 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0, and 20.0. A runtime error will be displayed if an illegal value is used. The data file need not be open before this method is called. A text entry will be made in the data file (if it has already been opened) indicating the lograte has changed.
Example
See Also `DataFileFormat.Close` (method), `DataFileFormat.Copy` (method), `DataFileFormat.FileUpdate` (method), `DataFileFormat.ForceEntry` (method), `DataFileFormat.IsOpen` (method), `DataFileFormat.Open`

(method), DataFileFormat.Snapshot (method), DataFileFormat.StartLogging (method), DataFileFormat.StopLogging (method) , DataFileFormat.WriteChannelHeader (method), DataFileFormat.WriteChannelNames (method), DataFileFormat.WriteChannelUnits (method)

DataFileFormat.Open (method)

Syntax DataFileFormat.Open *name$* [*, mode* [*, noheaderflag*]]

Description Used to open a data file object and prepare it for data logging.

Comments The *name$* parameter specifies the name of the disk file associated with this data file object. If the file already exists, the *mode* parameter specifies whether entries will be overwritten or entries will be added to the end of the file. Possible values for *mode* are Add or Overwrite. If the *mode* parameter is omitted, the default mode is Add. If the *noheaderflag* is omitted, a text entry will be made in the data file indicating a data file has been opened; if the *noheaderflag* is set to NoHeader or NH, no message is added to the data file. A runtime error will be displayed if the file cannot be opened for any reason.

Example

See Also DataFileFormat.Close (method), DataFileFormat.Copy (method), DataFileFormat.FileUpdate (method), DataFileFormat.ForceEntry (method), DataFileFormat.IsOpen (method), DataFileFormat.Lograte (method), DataFileFormat.Snapshot (method), DataFileFormat.StartLogging (method), DataFileFormat.StopLogging (method) , DataFileFormat.WriteChannelHeader (method), DataFileFormat.WriteChannelNames (method), DataFileFormat.WriteChannelUnits (method)

DataFileFormat.Snapshot (method)

Syntax DataFileFormat.Snapshot

Description Used to record an asynchronous sample of all channels being logged.

Comments The data file must be open before this method can be called. A runtime error will be displayed if the entry cannot be added to the data file.

Example

See Also DataFileFormat.Close (method), DataFileFormat.Copy (method), DataFileFormat.FileUpdate (method), DataFileFormat.ForceEntry (method), DataFileFormat.IsOpen (method), DataFileFormat.Lograte (method), DataFileFormat.Open (method), DataFileFormat.StartLogging (method), DataFileFormat.StopLogging (method) , DataFileFormat.WriteChannelHeader (method), DataFileFormat.WriteChannelNames (method), DataFileFormat.WriteChannelUnits (method)

DataFileFormat.StartLogging (method)

Syntax DataFile.StartLogging [*rate* [, *noheaderflag*]]

Description Used to start continuous, timed logging of selected channels to the data file.

Comments The optional *rate* parameter is a `Double` indicating the rate at which to log the data to the data file. Legal values for *rate* are 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0, and 20.0. The data file must be open before this method can be called. If the *noheaderflag* is omitted, a text entry will be made in the data file indicating that data logging has started and what the lograte is set to; if the *noheaderflag* is set to `NoHeader` or `NH`, no message is added to the data file. A runtime error will be displayed if the lograte is illegal or if the data file is not open.

Example

See Also `DataFileFormat.Close` (method), `DataFileFormat.Copy` (method), `DataFileFormat.FileUpdate` (method), `DataFileFormat.ForceEntry` (method), `DataFileFormat.IsOpen` (method), `DataFileFormat.Lograte` (method), `DataFileFormat.Open` (method), `DataFileFormat.Snapshot` (method), `DataFileFormat.StopLogging` (method) , `DataFileFormat.WriteChannelHeader` (method), `DataFileFormat.WriteChannelNames` (method), `DataFileFormat.WriteChannelUnits` (method)

DataFileFormat.StopLogging (method)

Syntax `DataFileFormat.StopLogging [`*noheaderflag*`]`

Description Used to stop timed data logging to the data file.

Comments The data file must be open and logging data before this method can be called. If the *noheaderflag* is omitted, a text entry will be made in the data file indicating that data logging has stopped; if the *noheaderflag* is set to `NoHeader` or `NH`, no message is added to the data file. A runtime error will be displayed if the data file is not open.

Example

See Also `DataFileFormat.Close` (method), `DataFileFormat.Copy` (method), `DataFileFormat.FileUpdate` (method), `DataFileFormat.ForceEntry` (method), `DataFileFormat.IsOpen` (method), `DataFileFormat.Lograte` (method), `DataFileFormat.Open` (method), `DataFileFormat.Snapshot` (method), `DataFileFormat.StartLogging` (method), `DataFileFormat.WriteChannelHeader` (method), `DataFileFormat.WriteChannelNames` (method), `DataFileFormat.WriteChannelUnits` (method)

DataFileFormat.WriteChannelHeader (method)

Syntax `DataFileFormat.WriteChannelHeader`

Description Used to write the list of channel names and channel units to the data file**.**

Comments The data file must be open before this method can be called. Accomplishes the same thing as successively calling `DataFileFormat.WriteChannelNames` and `DataFileFormat.WriteChannelUnits` in a TestPlan except this method guarantees that the channel names and channel units entries are logged in consecutive rows in the data file. A runtime error will be displayed if the entry cannot be added to the data file.

Example

See Also `DataFileFormat.Close` (method), `DataFileFormat.Copy` (method), `DataFileFormat.FileUpdate` (method), `DataFileFormat.ForceEntry` (method), `DataFileFormat.IsOpen` (method), `DataFileFormat.Lograte` (method), `DataFileFormat.Open` (method), `DataFileFormat.Snapshot` (method), `DataFileFormat.StartLogging` (method) , `DataFileFormat.WriteChannelNames` (method), `DataFileFormat.WriteChannelUnits` (method)

DataFileFormat.WriteChannelNames (method)

Syntax `DataFileFormat.WriteChannelNames`
Description Used to write the list of channel names to the data file.
Comments The data file must be open before this method can be called. A runtime error will be displayed if the entry cannot be added to the data file.
Example
See Also `DataFileFormat.Close` (method), `DataFileFormat.Copy` (method), `DataFileFormat.FileUpdate` (method), `DataFileFormat.ForceEntry` (method), `DataFileFormat.IsOpen` (method), `DataFileFormat.Lograte` (method), `DataFileFormat.Open` (method), `DataFileFormat.Snapshot` (method), `DataFileFormat.StartLogging` (method) , `DataFileFormat.WriteChannelHeader` (method), `DataFileFormat.WriteChannelUnits` (method)

DataFileFormat.WriteChannelUnits (method)

Syntax `DataFileFormat.WriteChannelUnits`
Description Used to write the list of channel units to the data file.
Comments The data file must be open before this method can be called. A runtime error will be displayed if the entry cannot be added to the data file.
Example
See Also `DataFileFormat.Close` (method), `DataFileFormat.Copy` (method), `DataFileFormat.FileUpdate` (method), `DataFileFormat.ForceEntry` (method), `DataFileFormat.IsOpen` (method), `DataFileFormat.Lograte` (method), `DataFileFormat.Open` (method), `DataFileFormat.Snapshot` (method), `DataFileFormat.StartLogging` (method) , `DataFileFormat.WriteChannelHeader` (method), `DataFileFormat.WriteChannelNames` (method)

ThisLimit.Disable (method)

Syntax `ThisLimit.Disable`
Description Behaves just like the `Limit.Disable` method.
Comments This method operates on the currently active limit that caused the limit handler to execute; whereas, `Limit.Disable` operates on a specific limit.
Example

ThisLimit.Enable (method)

Syntax `ThisLimit.Enable`
Description Behaves just like the `Limit.Enable` method.
Comments This method operates on the currently active limit that caused the limit handler to execute; whereas, `Limit.Enable` operates on a specific limit.
Example

ThisLimit.High (method)

Syntax *number* `= ThisLimit.High`
`ThisLimit.High` *number*
Description Behaves just like the `Limit.High` method.
Comments This method operates on the currently active limit that caused the limit handler to execute; whereas, `Limit.High` operates on a specific limit.
Example

ThisLimit.IsTripped (method)

Syntax *boolean* `= ThisLimit.IsTripped`
Description Behaves just like the `Limit.IsTripped` method.
Comments This method operates on the currently active limit that caused the limit handler to execute; whereas, `Limit.IsTripped` operates on a specific limit.
Example

ThisLimit.Low (method)

Syntax *number* `= ThisLimit.Low`
`ThisLimit.Low` *number*
Description Behaves just like the `Limit.Low` method.
Comments This method operates on the currently active limit that caused the limit handler to execute; whereas, `Limit.Low` operates on a specific limit.
Example

ThisLimit.Status (method)

Syntax *boolean* `= ThisLimit.Status`
Description Behaves just like the `Limit.Status` method.
Comments This method operates on the currently active limit that caused the limit handler to execute; whereas, `Limit.Status` operates on a specific limit.

VIP.Launch (method)

Syntax `VIP.Launch`
Description Used to run a Virtual Instrument Panel executable.
Comments
Example
See Also `VIP.Close` (method)

VIP.Close (method)

Syntax `VIP.Close`
Description Used to close a running Virtual Instrument Panel executable.
Comments
Example
See Also `VIP.Launch` (method)

Email.Address (property)

Syntax `Email.Address = "address"`
Description This property is used to set the destination address of an upcoming e-mail message.
Comments This property should be set prior to invoking the `SendMail` or `SendMailWithFile` methods.
Example `Email.Address = "dyno_supervisor@xyz_company.com"`

Email.SubjectLine (property)

Syntax `Email.SubjectLine = "text"`
Description This property is used to set the subject line of an upcoming e-mail message.
Comments This property should be set prior to invoking the `SendMail` or `SendMailWithFile` methods.
Example `Email.SubjectLine = "Message from Cell Assistant"`

Email.MessageLine (property)

Syntax `Email.MessageLine = "text"`
Description This property is used to set the message body of an upcoming e-mail message.
Comments This property should be set prior to invoking the `SendMail` or `SendMailWithFile` methods.
Example `Email.MessageLine = "Engine Test has passed."`

Email.AttachedFile (property)

Syntax `Email.AttachedFile = "filename"`
Description This property is used to specify the file that will be attached to the next e-mail message.
Comments This property should be set prior to invoking the `SendMailWithFile` method. It has no effect on the `SendMail` method.
Example `Email.AttachedFile = "d:\\ca4w\\DataFiles\\TestData.csv"`

Email.SendMail (Method)

Syntax `Email.SendMail`
Description This method causes an e-mail message to be sent.
Comments The sending address, subject line, and message line properties must be set prior to invoking this method.
Example As shown in Syntax.

Email.SendMailWithFile (Method)

Syntax `Email.SendMailWithFile`

Description This method causes and e-mail with an attached file to be sent.

Comments The sending address, subject line, message line and attached file properties must be set prior to invoking this method.

Example As shown in Syntax.

Appendix C - Computer Setup Instructions

TBD

Appendix D - CADAQ OLE Control

Prior to developing the DyneSystemsCo. VIPCONTROLS ActiveX control set, the only other way to create Virtual Instrument Panels (i.e.VIP's) has to purchase a control package from another vendor and use the DyneSystemsCo. CADAQ control to gain access to Cell Assistant channel data. While the CADAQ control is included in every release of Cell Assistant, it is no longer supported and/or updated. All future VIP development should be done using the VIPCONTROLS control set.

The sections that follow are an excerpt from previous Cell Assistant manuals that describe how to create a VIP using the CADAQ control, the standard Microsoft Visual Basic timer control, and a 3rd party analog meter control.

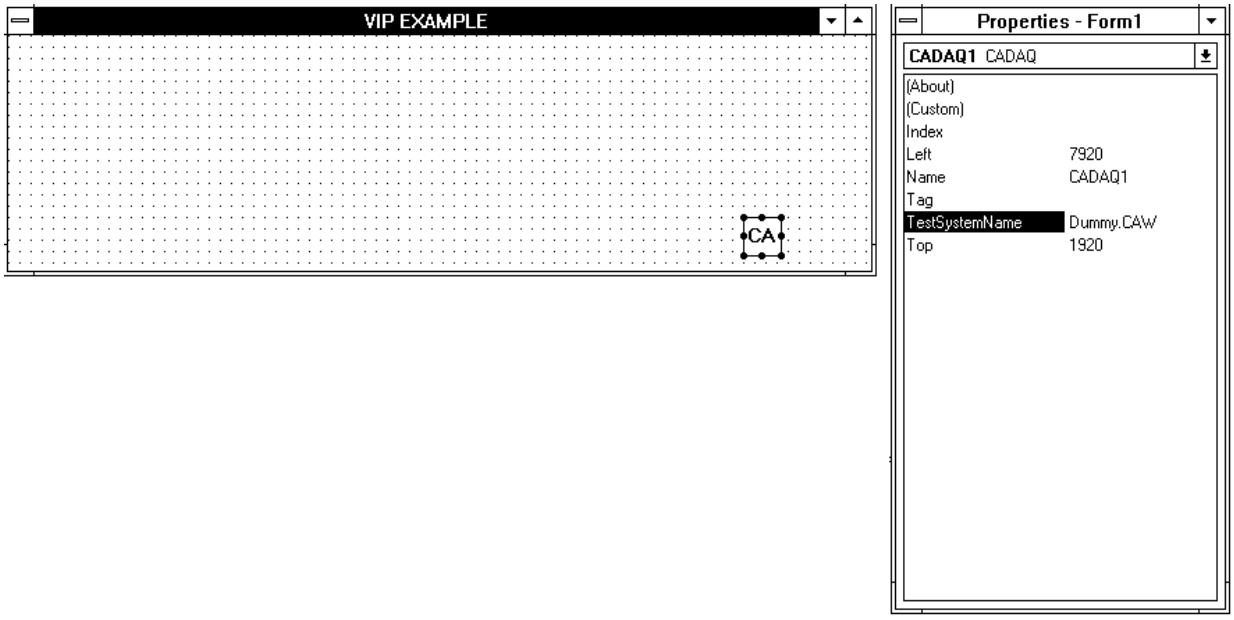
The excerpt begins …

Lets start developing our example VIP by placing the CADAQ (Cell Assistant Data Acquisition Control) OLE control on our VB form. To do this, click on the VB toolbox button that has the red "CA". Then click somewhere on the form, keeping the mouse button pressed and dragging a rectangle on the form. When the button is released, the CADAQ object will appear. It's not important where the object is placed on the form because it will not be visible when the VIP runs.

Next you need to modify some of CADAQ's properties. View the property box by selecting the CADAQ object (handles in the corners and on the sides will show when an object is selected) and pressing F4 on your keyboard. There are two critical properties that must be addressed for this control to work properly. The first one is the "TestSystemName" property, and the second one is the "Name" property.

The "TestSystemName" property refers to the TestSystem in Cell Assistant that is currently open, or the one that will be used with the VIP you are developing. The TestSystem name that we are using for our example is Dummy.CAW.

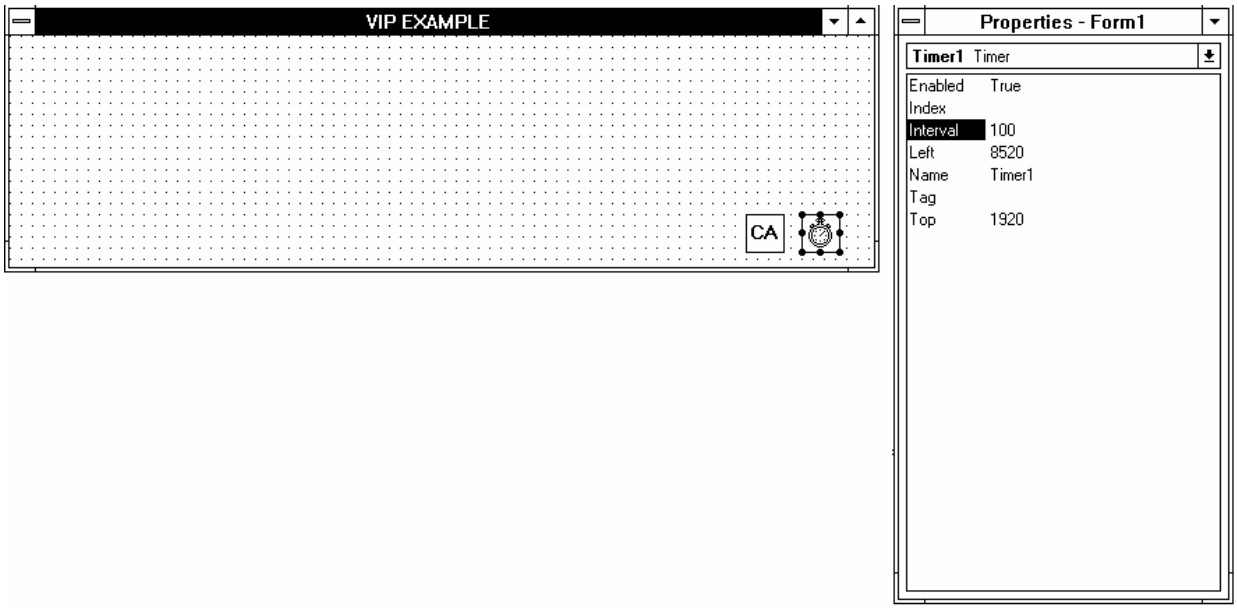
The name property, as you might expect, refers to the name you assign the CADAQ control. For our example VIP, we will leave the default name CADAQ1 as the name for the CADAQ OLE control.



The Timer Control:

Next lets place a standard VB timer control on our form. The properties box for the timer control should appear after you have placed the control on the form. The timer event will be used to update our virtual instruments (gages, buttons etc.) on a timed basis. We will enter code under the timer event later on in this document. The "Interval" property used on the timer control determines the interval at which the timer event code will get executed. Set this interval for 100ms, which will update our controls at a rate of ten times a second, or 10 Hz.

For our example, we will leave the name of the timer control as Timer1, which is the VB default for the first timer on the form. Make sure the "Enabled" property is set to "True".



The VICOMponents™ Analog Meter:

Now we will add a Computer Boards VICOMponents™ analog meter to our form. Assuming you have installed the VICOMponents™ software on your computer so that the controls are available to you in the VB environment, place a analog meter control on the VB form. Notice that the property box changes to AnalogMeter1. For our example, change the "Name" property to "SpeedGage". The property box should reflect the name change of the control. Next, highlight the "Custom" property and notice that a button appears to the right of the list box. Click on this button to see the custom property dialog for the control. This tabbed dialog box allows us to customize our knob control.

Click on the tab labeled "Scale". This is where we set the scale on our dial. For our example, set the minimum value to 0, and the maximum value to 3000.

VIP Code

Now that we have configured our VIP form and controls for our VIP, lets write some code to make our VIP operational. The following lines of VB code provide our VIP with a means to access and display channel information from within our VIP application. The Dyne Systems CADAQ control that we configured for our VIP provides us with a "link" between our VIP and the TestSystem channel information.

Lets begin by declaring some "Public" handles in the "General" declarations section in VB. We will create one handle per control. In addition, we will declare an array variable because this is the form of variable that VICOMponents™ needs.

(**General**)

```
Public h_Speed As Integer
Dim fValue (0 to 3) As Single
```

Next, lets go to the Form Load event procedure and write some code that will open the TestSystem and register the channels we will be using in our VIP:

```
CADAQ1.OpenTestSystem("Dummy.CAW")
h_Speed  =  CADAQ1.RegisterChannel("aicSpeed")
```

The first line of code calls the CADAQ to open a TestSystem called "Dummy.CAW".

Next, we "register" channels from our TestSystem using the CADAQ's RegisterChannel method. The RegisterChannel method contains one argument enclosed in parentheses and quotes. The argument is the name of a Cell Assistant channel used in our TestSystem that provides data that is of use to our VIP. Finally, by setting one of the handles we created equal to a registered channel, we have provided a means for VB to refer to a specific channel in the TestSystem.

Next, enter the following code into the Timer1 timer event procedure.

Timer1

```
fValue(0) = CADAQ1.GetDataAnalog (h_Speed)
SpeedGage.SetValue fValue(0)
```

The Timer event procedure updates the controls in our VIP periodically. The “interval” property for the timer sets the interval at which the above code will be executed. The interval value is in milliseconds.

The first line in our timer event uses the CADAQ.GetDataAnalog method to retrieve data from the TestSystem. There is one argument for the GetDataAnalog method - the handle name (enclosed in parenthesis). The handle argument contains the handle we created that is registered to the channel that contains the data needed for our gage. In this case, the handle is h_Speed. The first element of the fValue array (fValue(0)) is set equal to the CADAQ1.GetDataAnalog(h_Speed) statement. In the next line, the SetValue property for our gage is set equal to the first element of the fValue array (fValue(0)). See your VICOMponents™ manual for a list of properties and methods for their controls.

A few suggestions before you proceed to debugging the VIP. Scan your code for blatant syntax errors (VB will usually alert you if this is the case...99% of the time). Recheck the spelling of channel names called out in your code to make sure they match the spelling of the channel names in your TestPlan.

Make sure data acquisition or a valid TestPlan (valid refers to the one you entered in the CADAQ TestSystemName property and in the Form Load event) is running.

AND ALWAYS make sure that commands coming from your VIP, intentional or unintentional will not cause something to happen that will cause damage to the devices or machinery they ultimately control, or cause injury or unsafe conditions for the folks working in or around the area.

… the excerpt ends.

Index

1

1 Point calibration, 9-1
1/10 LAC, 4-13

2

2 Point calibration, 9-2

A

About CA4W, 3-8
Add Procedure, 6-2
Adding
  Channels, 5-2
  Hardware Device, 4-1
Allen Bradley SLC 5/03 PLC
  Channels, A-1
  Port Setup, 4-1
  Settings, 4-2
  Setup, 4-1
Analog Input Channel, 5-4
Analog Output Channel, 5-5
Application Name, 5-17
Arrange Icons, 3-8
ASCII communication, 4-9
ASCII File
  Channels, A-3
  Diagnose, 4-4
  Settings, 4-3
  Setup, 4-3
Autozero property, 5-5
Average Channel, 5-6
AVL 439 Opacimeter
  Channels, A-7
AVL Fuel Balance
  Channels, A-5
  Diagnose, 4-5
  Port Setup, 4-4
  Setup, 4-4

B

Baldor Series 22H Vector Drive
  Channels, A-9
  Diagnose, 4-6
  Port Setup, 4-5
  Settings, 4-5
  Setup, 4-5
Baldor Series 22H Vector Drive Diagnose, 4-6
BEE, 4-8
Binary Communications, 4-9
Binary Set Points, 4-13

C

CADAQ Control, D-1
CAI Emissions Bench
  Channels, A-11
Calculated Channels, 5-6
Calibration, 9-1
  1 Point, 9-1
  2 Point, 9-2
  Manual, 9-2
  Restoring, 9-3
  Saving, 9-2
Calibration File, 9-3
Calibration Level, 3-2
Cascade, 3-8
Channel Data
  View, 3-7
Channel List, 5-1
Channel Property, 6-4
Channels, 5-1
  Adding, 5-2
  Analog Input, 5-4
  Analog Output, 5-5
  Average, 5-6
  Calculated, 5-6
  Clipping Channel Names, 5-3
  Configuring, 5-3
  Counter, 5-14
  DDE Input, 5-17
  Device Control Output, 5-15
  Device Status Input, 5-15
  Digital Input, 5-15
  Digital Output, 5-16
  Removing, 5-2
  Renaming, 5-2
  Text Input, 5-17
  Text Output, 5-17
  Timer, 5-17
Close Method, 8-5
Companion
  Channels, A-13
  Configure, 4-7
  Diagnose, 4-7
  Port Setup, 4-6
  Settings, 4-7
  Setup, 4-6
Computed Channels, 5-4

Computer Setup, 2-1
Configuration Level, 3-2
Configuring Channels, 5-3
Control Main Menu Item, 3-8
Copy Method, 8-6
Copy Protect Block, 2-2
Correction Factor
  ISO 3046-1, 5-8
  J1349, 5-7
  J607, 5-7
Counter Channel, 5-14
  Use In TestPlan, 5-15
CurrentLimitSet Property, 6-8

D

Data Acquisition
  Start and Stop, 3-8
Data button
  Daytronic System 10 Diagnose, 4-9
  DTC Diagnose, 4-12
  Dyn-Loc/DTC Diagnose, 4-14
  Pierburg Emissions Bench, 4-16
  Sartorius LC3201D Diagnose, 4-18
Data Conversion, 5-4
Data File
  View, 3-7
Data File Formats
  Adding, 8-1
  Configuring, 8-1
  File Sharing, 8-7
  Removing, 8-1
  Renaming, 8-1
  Using in TestPlans, 8-2
Data Files, 8-1
  Application for Viewing, 3-5
  Close Method, 8-5
  Copy Method, 8-6
  FileUpdate Method, 8-7
  ForceEntry Method, 8-5
  IsOpen Method, 8-5
  Location, 8-3
  LogRate Property, 8-4
  Open Method, 8-2
  Snapshot Method, 8-4
  StartLogging Method, 8-3
  StopLogging Method, 8-4
  WriteChannelHeader Method, 8-6
  WriteChannelNames Method, 8-6
  WriteChannelUnits Method, 8-6
Daytronic System 10
  Channels, A-15
  Configure, 4-8
  Diagnose, 4-9
  Port Setup, 4-7
  Settings, 4-8
  Setup, 4-7
DDE Input Channel, 5-17
Decimal Places, 5-4
Default TestPlan, 5-7, 5-12, 7-2
Device Control Output Channel, 5-15
Device Sataus Input Channel, 5-15
Digital Input Channel, 5-15
  Use In TestPlan, 5-16
Digital Output Channel, 5-16
  Use In TestPlan, 5-16
Direct Logic 205 PLC
  Channels, A-16
  Port Setup, 4-10
  Settings, 4-10
  Setup, 4-9
Disable Method, 6-3
Display Filter, 5-1
Division by Zero, 5-14
Download
  Daytronic System 10, 4-9
Drivers Trace
  Channels, A-18
DTC
  Channels, A-19
  Diagnose, 4-12
  Port Setup, 4-12
  Setup, 4-12
Dyn-Loc/DTC
  Channels, A-21
  Diagnose, 4-14
  Port Setup, 4-13
  Settings, 4-13
  Setup, 4-13

E

Edit Main Menu Item, 3-7
EEPROM, 4-9
E-Mail, 3-4
EMM, 4-8
Enable Method, 6-3
Engineering Units, 5-4
EngUnits.txt, 5-4
EQ, 12-4

F

FE data line type, 8-5
FIL, 4-8
File Main Menu Item, 3-2
File Sharing, 3-4
FileUpdate Method, 8-7
Filter
  Daytronic System 10, 4-8
Flight Recorder File
  View, 3-7
Flight Recorder Properties, 7-4
Font Selection, 3-5
ForceEntry Method, 8-5
Function Property, 6-4

G

Global TestPlan, 5-7, 5-12, 7-2

H

Hardware Device
  Adding, 4-1
  Removing, 4-1
  Setup, 4-1
HD data line type, 8-3, 8-4, 8-5
Help, 3-8
Help Index, 3-8
Help Main Menu Item, 3-8
High Property, 6-4
Horiba EXSA-1500 (GPIB Version)
  Channels, A-23

I

Installation, 2-1
Invert Logic, 5-15, 5-16
IsOpen Method, 8-5
IsTripped Property, 6-5
Item Name, 5-17

L

Limit Handler Format, 6-5
Limit Handler Return Value, 6-6
Limit Sets
  View, 3-7
Limit Sets Page, 6-7
Limits, 6-1. *See* Limits
  Channel Property, 6-4
  CurrentLimitSet Property, 6-8
  Disable Method, 6-3
  Enable Method, 6-3
  Function Property, 6-4
  High Property, 6-4
  IsTripped Property, 6-5
  Limit Handler Format, 6-5
  Limit Handler Return Value, 6-6
  Limit Sets, 6-6
  Limit Sets Page, 6-7
  Low Property, 6-3
  Monitored Channel, 6-1
  Priority, 6-2
  Procedure, 6-2
  Status Property, 6-4
  The DSBasic Limit handler, 6-5
  ThisLimit object, 6-6
  Trigger Type, 6-2
  Type, 6-1
  Using in TestPlans, 6-3
LM data line type, 8-2
Log Out, 3-2
LogRate Property, 8-4
Low Property, 6-3

M

Manual calibration, 9-2
Mini-Companion
  Channels, A-25
Modicon PLC
  Channels, A-26
  Port Setup, 4-14
  Settings, 4-15
  Setup, 4-14
Modify
  Daytronic card, 4-8
Most Recently Used File List, 3-4
Most Recently Used Files, 3-4
Move Down button, 4-8
Move Up button, 4-8

N

Network VIP's, 3-4

O

OFF Text*, *5-15, 5-16
Offset
  Daytronic System 10, 4-8
Offset b, 5-4
ON Text*, *5-15, 5-16
Open Method, 8-2

P

Password, 3-2
  Changing, 3-2
  Entering, 3-2
Password Menu Item, 3-3
Paste Link, 5-17
Physical Channels, 5-4
Pierburg CVS
  Channels, A-28
Pierburg Emissions Bench
  Channels, A-30
  Diagnose, 4-16
  Port Setup, 4-16
  Setup, 4-16
Pierburg PDP CVS
  Diagnose, 4-17
  Port Settings, 4-17
  Setup, 4-17
Polled, 5-3
Preferences, 3-4

Preferences Menu Item, 3-3
Preset method, 5-15, 5-18
Print Preview Menu Item, 3-3
Print Setup Menu Item, 3-3
Printing
TestPlans, 3-3
TestSystems, 3-3

R

Recently Used File List, 3-3
Removing
Channels, 5-2
Hardware Device, 4-1
Renaming
Channels, 5-2
Reset method, 5-15, 5-18
Reset property, 5-6
Restoring calibrations, 9-3
Rosemount Micro Motion RFT9712 Fuel Meter
Channels, A-33

S

Sample Frequency
Allen Bradley SLC 5/03 PLC, 4-2
Direct Logic 205 PLC, 4-11
Dyn-Loc, 4-14
Modicon PLC, 4-15
Sartorius LC3201D, 4-18
Sample Rate, 5-3
Sartorius LC3201D
Channels, A-34
Diagnose, 4-18
Setup, 4-17
Sartorius LC3210D
Port Setup, 4-17
Saving calibrations, 9-2
Send button
Baldor Series 22H Vector Drive Diagnose, 4-6
DTC diagnose, 4-12
Dyn-Loc/DTC diagnose, 4-14
Pierburg Emissions Bench diagnose, 4-16
Pierburg PDP CVS diagnose, 4-17
Sartorius LC3201D diagnose, 4-18
Set Point Multipliers, 4-13
Setup
Hardware Device, 4-1
Shift Register, 5-6
Shortcut, 3-1
Show Raw Data, 5-4
SL data line type, 8-5
Slave Address
Direct Logic 205 PLC, 4-11
Modicon PLC, 4-16
Slope
Daytronic System 10, 4-8
Slope m, 5-4
Snapshot Method, 8-4
Start method, 5-15, 5-18
StartLogging Method, 8-3
Station Number
Allen Bradley SLC 5/03 PLC, 4-2
Status Bar, 3-9
Status Property, 6-4
Stop Data Acquisition, 7-2
Stop method, 5-15, 5-18
StopLogging Method, 8-4
Symbol Technologies LS1004
Channels, A-36
SYSTEM Device
Channels, A-37
System Function, 5-7
System Functions, 5-7

T

Test Application Property Page, 3-5
TestPlan, 7-1
Running, 3-8
Stopping, 3-8
TestPlan editor options, 3-5
TestPlan Functions, 5-12
TestPlan Stop, 7-2
TestPlan stopping, 3-8
TestPlans, 7-1
TestSystem, 3-1
Text Input Channel, 5-17
Text Output Channel, 5-17
Text property, 5-16
ThisLimit object, 6-6
Timer Channel, 5-17
Use In TestPlan, 5-18
TL data line type, 8-4
Tool Bar, 3-8
Tool Bar Buttons, 3-8
Topic Name, 5-17

U

Units Property, 5-5, 5-6
User Level, 3-2

V

Value Property, 5-5, *5-6*, 5-16
View Interface Property Page, 3-5
View Main Menu Item, 3-7
VIP, 10-1
VIP Interface
Setup, 4-18
VIPCONTROLS
AnalogMeter, 10-4
BarMeter, 10-4

ComputerSelect, 10-6
DigitalMeter, 10-4
Meter Borders, 10-7
PilotLight, 10-4
PushButton, 10-5
Selecting a Channel, 10-6
StripChart, 10-5
TextMeter, 10-4
Using Channel Properties, 10-6
Using Limit, 10-7
Virtual Instrument Panel Driver
Channels, A-38
Virtual Instrument Panels, 10-1
Adding VIP's, 10-1
Creating VIP's, 10-2
Removing VIP's, 10-2
Renaming VIP's, 10-2
Using in TestPlans, 10-2
Using VIP's, 10-1
VIPCONTROLS ActiveX Control, 10-4
Visual Basic, 10-1
Location, 3-5

W

Watchdog, 3-6
Window Main Menu Item, 3-8
WriteChannelHeader Method, 8-6
WriteChannelNames Method, 8-6
WriteChannelUnits Method, 8-6

X

XM Series Gas Analyzer
Channels, A-39